Towards new methods of clustering data stream

Mohammed Oualid Attaoui

▶ **To cite this version:**

Mohammed Oualid Attaoui. Towards new methods of clustering data stream. Databases [cs.DB]. Université Paris-Nord - Paris XIII; Ecole Nationale Supérieure d'Informatique (ESI) - Alger, 2021. English. NNT : 2021PA131082 . tel-03969236

HAL Id: tel-03969236

https://theses.hal.science/tel-03969236

Submitted on 2 Feb 2023

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.

N°d'ordre: .....

Université Sorbonne Paris Nord
Laboratoire LIPN
Ecole Supérieure en Informatique
Laboratoire LabRI

Thèse de Doctorat en Informatique

Filière : Informatique
Spécialité : Système d'information

Par

Mr Mohammed Oualid ATTAOUI

Vers de nouvelles méthodes de clustering de flux de données

Soutenue le 30 Juin 2021 devant le jury :

Pr.	Sidi Mohammed BENSLIMANE	ESI de Sidi Bel-Abbes, LabRI Algérie	Président du jury
Dr.	Djamel BOUCHAFFRA	CDTA Alger, Algérie	Examinateur
Dr.	Cyril DE RUNZ	Université de Tours, France	Examinateur
Pr.	Thierry CHARNOIS	Université Sorbonne Paris Nord LIRIS, France	Examinateur
Pr.	Germain FORESTIER	Université d'Haute-Alsace, IRIMAS, France	Invité
Dr.	Hanene AZZAG	Université Sorbonne Paris Nord, LIPN, France	Invitée
Dr.	Mustapha LEBBAH	Université Sorbonne Paris Nord, LIPN, France	Directeur de thèse
Pr.	Nabil KESKES	ESI SBA, LabRI, Algérie	Co-Directeur de thèse

Année Universitaire : 2020 - 2021

À mes Parents, Frère, Sœurs, et Èpouse...

ملخص

في الأيام الأخيرة ، يتم إنشاء كميات كبيرة من البيانات بواسطة التطبيقات في الوقت الفعلي. لا يمكن التعامل مع هذه الكميات من البيانات التي تسمى تدفقات البيانات مثل البيانات العادية لأننا لا نستطيع تخزين أو معالجة هذه الكمية من البيانات. تعدين التدفق هو عملية إيجاد بنية معقدة في حجم كبير من البيانات حيث تنتقل البيانات وتصل في دفق غير مقيد. تدفق البيانات هو تسلسل مستمر من البيانات يفرض قيود مرور واحدة. الوصول العشوائي إلى البيانات غير ممكن ، ومن غير العملي تخزين جميع البيانات التي تصل. في هذه الحالة ، نقوم بتخزين خصائص أو ملخصات المجموعات التي عادةً ما تتضمن إحصائيات وصفية للعنقود.

في كثير من الحالات ، يجب أن تحترم خوارزميات تدفق البيانات قيود المكان والزمان. غالبًا ما تحتوي البيانات التي تصل إلى التدفقات على ضوضاء وقيم متطرفة. وبالتالي ، يجب أن يقوم تجميع تدفق البيانات باكتشاف هذه البيانات وتمييزها وتصفيتها قبل مهمة التجميع. يركز العمل الحالي على نمذجة البيانات عالية الأبعاد في إطار عمل تدفق البيانات ، وذلك باستخدام مجموعة الفضاء الجزئي لاكتشاف مجموعات متكاملة في فضاءات فرعية مختلفة. استخدمنا أيضًا إطار العمل متعدد الأهداف للتعامل مع الاختلافات في خصائص البيانات. لقد قدمنا تقنيات مختلفة تعتمد على تجميع الفضاء الجزئي ، والتكتل متعدد الأغراض جنبًا إلى جنب مع تقنيات تحليل التدفق لمعالجة المشكلات المذكورة سابقًا.

لقد قدمنا طريقة لتجميع الفضاء الجزئي بنموذجين للسمات والكتلة، وهي طريقة فعالة لتجميع الفضاء الجزئي لتيار بيانات قابل للتطوير بطريقة مضمنة. استخدمنا الأوزان التي تم الحصول عليها كدرجات لمزيد من التجارب. تم استخدام طريقة تجميع الفضاء الجزئي مع نموذج الترجيح العالمي كطريقة لتقليل الأبعاد. لقد أظهرنا تأثير ترتيب نقاط البيانات وتداخل النافذة على جودة التجميع لقد قدمنا طريقة جديدة لتجميع تدفق البيانات بناءً على خوارزمية متعددة الأهداف. والذي يستخدم وظيفتين موضوعيتين لإيجاد مجموعات ذات شكل تعسفي وتحسين جودة التجميع. تستخدم هذه الخوارزمية عملية من مرحلتين : ١) مرحلة في الوقت الحقيقي : إنشاء العديد من حلول التجميع بناءً على خوارزميات مختلفة وعوامل جينية ٢) مرحلة التجميع النهائي : إنشاء قسم مثالي من المجموعات المكتشفة. طبقنا طريقتنا على مجموعات بيانات التدفق الكبيرة وقارناها بخوارزمية أخرى لتجميع التدفق. لقد قدمنا تحسينًا للطريقة السابقة. تعمل الطريقة الجديدة على تحسين وقت الحساب باستخدام وقت التوقف لتحسين الحل. كما أنه يحسن تخصيص الذاكرة من خلال قديم نهج تجميع شجرة جديد لتخزين موجز للبيانات فقط بدلاً من البيانات بأكملها. أخيرًا ، قدمنا طريقة جديدة لاستخراج وتجميع البقع ذات الصلة من الصور النسيجية المرضية. تجمع هذه الطريقة بين تجميع الفضاء الجزئي والتقنيات متعددة الأهداف للتغلب على مشكلة البقع غير ذات الصلة. يهدف نهجنا إلى تحديد المواقع ذات الصلة بالتصنيف بدلاً من استخدام الصورة بأكملها أو جميع المناطق المحددة باستخدام طريقة النافذة المنزلقة. لقد قدمنا نسختين من الطريقة. يتعامل الإصدار الأول مع مجموعات البيانات الرقمية ، وقد تم اختبار أدائها على مجموعات البيانات التركيبية. يتعامل الإصدار الثاني مع مجموعات بيانات غجـ التي قمنا بتكييفها لتكون متوافقة مع التجميع. يسمح لنا هذا الإصدار باستخراج التصحيحات ذات الصلة أثناء تجميعها في مجموعات متشابهة.

Résumé

Ces derniers jours, de grandes quantités de données sont générées par les applications en temps réel. Ces quantités de données appelées flux de données ne peuvent pas être traitées comme des données classiques car nous ne pouvons pas stocker ou traiter cette quantité de données. L'exploration de flux est le processus qui consiste à trouver une structure complexe dans un grand volume de données où les données évoluent et arrivent dans un flux non limité. Un flux de données est une séquence de données continues qui impose une restriction de passage unique. L'accès aléatoire aux données n'est pas possible, et il est peu pratique de stocker toutes les données qui arrivent. Dans ce cas, nous stockons des caractéristiques ou des synopsis de clusters qui comprennent généralement des statistiques descriptives pour un cluster. Dans de nombreux cas, les algorithmes de flux de données doivent respecter des contraintes d'espace

et de temps. Les données arrivant dans les flux contiennent souvent du bruit et des valeurs aberrantes. Ainsi, le clustering de flux de données doit détecter, distinguer et filtrer ces données avant la tâche de clustering. Le présent travail porte sur la modélisation de données à haute dimension dans un cadre de flux de données, en utilisant le Subspace Clustering pour découvrir des clusters intégrés dans différents sous-espaces. Nous avons également utilisé le cadre multi-objectif pour faire face aux variations des caractéristiques des données. Nous avons présenté différentes techniques basées sur le subspace clustering, le Multi-Objective clustering combiné avec des techniques d'analyse de flux pour répondre aux problèmes mentionnés précédemment.

Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse, M. Mustapha Lebbah pour m'avoir accueilli au sein de son équipe. Je lui suis également reconnaissant pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

J'adresse de chaleureux remerciements à mon co-encadrant de thèse, M. Nabil Keskes, pour son attention de tout instant sur mes travaux. Son énergie et sa confiance ont été des éléments moteurs pour moi.

Un grand merci à Mme Henene Azzag pour son implication dans le projet, pour ses conseils avisés et son écoute qui ont été prépondérants pour la bonne réussite de cette thèse. Elle m'a beaucoup appris, j'ai apprécié son enthousiasme et sa sympathie. J'ai pris un grand plaisir à travailler avec elle.

Je voudrais remercier les membres du jury de cette thèse M. Sidi Mohammed Benslimane, M. Djamel Bouchaffra, M. Cyril De Runz, M. Thierry Charnois et M. Germain Forestier, pour l'intérêt qu'ils ont porté à mon travail et leurs conseils pour l'améliorer.

J'associe à ces remerciements mon tuteur M. Joseph Ben Gelloun qui m'a donné des conseils avisés et a été une grande ressource pour l'accomplissement de ce travail .

Je souhaite remercier spécialement ma femme Aya pour son soutien et sa patience tout au long de la thèse. Pour notre futur petite fille Sarah.

Je remercie mes parents, mon frère et mes sœurs, Abbes, Sarah, Ikram et Maroua, pour leur soutien au cours de ces trois années et sans lesquels je n'en serais pas là aujourd'hui.

Je remercie mes collègues a l'université Sorbone Paris Nord, Gaël, Anthony, Florent, Antoine, Etienne, Dina, Massinissa, Mohammed. Je remercie aussi tout mes collègues à l'école superieure en informatique de SBA.

Paris, le 30 Juin 2021.

Contents

List of Figures

List of Tables

General Introduction 1

1.1 Problem and Context

The French philosopher Michel Foucault in his book "Les mots et les choses: Une archéologie des sciences humaines" [Foucault, 1966] affirms that the human representation of the world, the "fundamentals codes of a culture," establish a "system of elements," a kind of set of rules that allows individuals to make sense of the world by finding similarities and differences between elements according to some pattern.

Recently, the development of the information society has led to the acquisition and management of extensive collections of data described in high dimensional spaces. Significant efforts have been made to develop automatic tools, such as clustering, that could help find some order to these datasets and better grasp the complexity they convey.

The fundamental principles of the clustering algorithms are reminiscent of the ideas of Foucault: Clustering aims to partition a set of objects into clusters such that :

- Objects, in the same cluster, must be similar as much as possible
- Objects, in the different clusters, must be different as much as possible

However, in recent days, large amounts of data are generated by applications in real-time. Figure (1.1) shows the amount of mobile data traffic around the world in a period of 10 years. We notice how the amount of data grows exponentially especially after 2019 with the appearance of 5G.

This amounts of data called stream data cannot be processed like classic data since we can not store or process this amount of data. Stream mining is the process of finding a complex structure within a large volume of data where the data evolves and arrives in an unbounded stream. A data stream is a sequence of continuous data that imposes a single pass restriction. Random access to the data is not feasible, and it is impractical to store all the arriving data. In this case, we store cluster features or synopses that typically include descriptive statistics for a cluster. In many cases, data stream algorithms have to observe space and time constraints. Data arriving in streams often contain noise and outliers. Thus, data stream clustering should detect, distinguish, and filter this data before the clustering task.

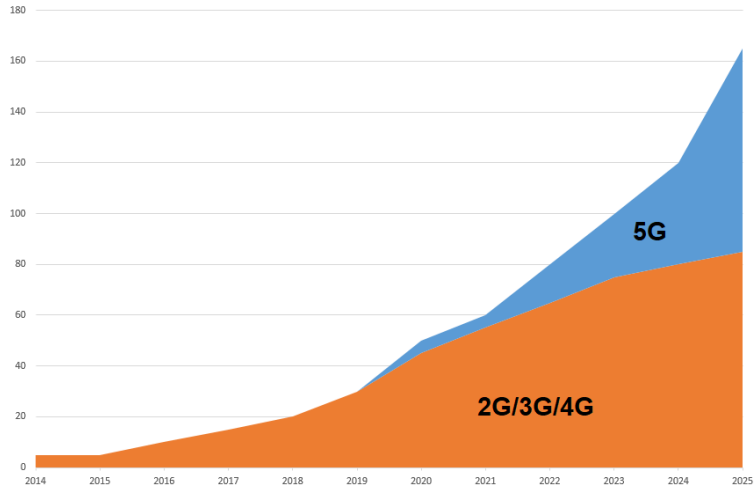


Figure 1.1 – *Global mobile data traffic (EB per month).*

1.2 Challenges

In CERN, the European Organization for Nuclear Research, Experiments are generating an entire petabyte (10^6 GB) of data every second as particles fired around the Large Hadron Collider[1]. *« We don't store all data as that would be impractical. Instead from the collisions, we run, we only keep a few pieces that are of interest, the rare events that occur, which our filters spot and send over the network »*[2]. In other terms, CERN stores 25PB of data every year; this amount of data should be analyzed to find patterns that can help to understand the structure and make-up of the universe. Even though memory capacity has quadrupeled every 3 years since its creation, the data grows faster than any memory capacity we have (1.1) and it causes many issues, we present some of them in the following:

- Data velocity: Data stream evolves and arrives in an unbounded stream. A data stream is a sequence of continuous data that imposes a single pass restriction. Random access to the data is not feasible.
- High-dimensional data streams: recent advances in data acquisition do not only imply that the objects may be described in high-dimensional spaces. This leads to traditional clustering techniques to struggle when dealing with high dimensional datasets.
- Processing time: Data stream evolves with high speed. The incoming data stream should be processed on-the-fly.
- Memory restrictions: As mentioned above, data is growing faster than our capacity to store it. Storing these large amounts of data is not possible.
- Multi-view datasets: Data can be collected from multiple sources or multiple facets. In such setting, each point is associated with much richer information

[1] https://home.cern/science/computing
[2] https://www.computing.co.uk/?v3

1.3 Contributions

We presented different techniques based on subspace clustering, Multi-Objective clustering combined with stream analysis techniques to respond to the previously mentioned issues. Our contributions are resumed below:

- We presented an extended state of the art on clustering static data and stream data. We provided comparisons between clustering categories and clustering algorithms. We also presented the techniques and concepts of clustering classic and stream data and the main datasets, frameworks, open-source resources, etc.
- We presented a subspace clustering method with two models of feature and block weighting (global and local), an efficient method for subspace clustering of an evolving data stream in an online manner. We used the weights obtained as scores to conduct more experiments. The subspace clustering method with the Global Weighting Model was used as a dimensionality reduction method. We proved the impact on the order of the data point and the windows' overlapping to the clustering quality.
- We presented a new clustering data stream method based on a multi-objective algorithm called MOC-Stream that employs two objective functions to find arbitrary shaped clusters and enhance the clustering quality. MOC-Stream uses a two-phase process: 1) online phase: creating several clustering solutions based on different algorithms and genetic operators 2) offline phase: construction of an optimal partition from the discovered clusters. We applied our method on large stream datasets and compared it to a different stream clustering algorithm.
- We presented an improvement for the previous method. The new method optimizes the computation time by using idle times to improve the solution. It also optimizes memory allocation by introducing a new tree aggregation approach for the Ant-Tree algorithm to store only a synopsis of the data instead of all the dataset.
- Finally, we presented a new method of extracting and clustering relevant patches from histopathological images. It combines subspace clustering and multi-objective techniques to overcome the problems of non-relevant patches. Our approach aims to select the relevant patches for classification instead of using the whole image or all selected patches using the sliding window method. We presented two versions of the method. The first one deals with numerical datasets, and was tested its performance on synthetic datasets. The second version deal with RGB datasets that we adapted to be compatible with the clustering. This version allows us to extract the relevant patches while grouping them into similar clusters.

1.4 Outline

The present work is concerned with the modelling of high-dimensional data within a data streaming framework, using Subspace clustering to discover clusters embedded in different subspaces. We also used the Multi-objective framework to cope with the variations in the data characteristics. The subsequent sections are organised as follows:

1. in **Chapter 3**: we introduce the main clustering categories and compare between them. We also give an overview of the main algorithm of each category and another comparison between these algorithms.

2. in **Chapter 4** we discuss the clustering of stream data and its main properties and techniques. We outline an overview of the clustering stream data methods and provide a comparison between these methods.

3. in **Chapter 5** we present our first approach, S2G-Stream with two models of feature and block weighting (Global and Local), an efficient method for subspace clustering of evolving data stream in an online manner.

4. in **Chapter 6**, our second approach is presented with two variants MOC and IMOC. These approaches employ two objective functions to find clusters of arbitrary shaped clusters and enhance the clustering quality.

5. in **Chapter 7** we discuss our third approach SMO-HPS. The method combines subspace clustering and multi-objective techniques to extract relevant patches from histopathological images and cluster them.

Introduction Générale 2

Cette introduction est la version en français de l'introduction précédente.

2.1 Problème et Contexte

Le philosophe français Michel Foucault, dans son ouvrage "Les mots et les choses : Une archéologie des sciences humaines" [Foucault, 1966] affirme que la représentation humaine du monde, les "codes fondamentaux d'une culture", établissent un "système d'éléments", une sorte d'ensemble de règles qui permet aux individus de donner un sens au monde en trouvant des similitudes et des différences entre les éléments selon un certain pattern.

Récemment, le développement de la société de l'information a conduit à l'acquisition et à la gestion de vastes collections de données décrites dans des espaces à haute dimension. Des efforts importants ont été déployés pour développer des outils automatiques, tels que le clustering, qui pourraient aider à mettre de l'ordre dans ces ensembles de données et à mieux appréhender la complexité qu'ils représentent.

Les principes fondamentaux des algorithmes de clustering rappellent les idées de Foucault : Le clustering vise à partitionner un ensemble d'objets en clusters tels que :

- Les objets, dans le même cluster, doivent être similaires autant que possible.
- Les objets, dans les différents clusters, doivent être différents autant que possible

Cependant, ces derniers jours, de grandes quantités de données sont générées par les applications en temps réel. La figure (1.1) montre le volume du trafic de données mobiles dans le monde sur une période de 10 ans. Nous remarquons comment la quantité de données augmente de manière exponentielle surtout après 2019 avec l'apparition de la 5G. Ces quantités de données appelées flux de données ne peuvent pas être traitées comme des données classiques car nous ne pouvons pas stocker ou traiter cette quantité de données. L'exploration de flux est le processus qui consiste à trouver une structure complexe dans un grand volume de données où les données évoluent et arrivent dans un flux non limité. Un flux de données est une séquence de données continues qui impose une restriction de passage unique. L'accès aléatoire aux données n'est pas possible, et

il est peu pratique de stocker toutes les données qui arrivent. Dans ce cas, nous stockons des caractéristiques ou des synopsis de clusters qui comprennent généralement des statistiques descriptives pour un cluster. Dans de nombreux cas, les algorithmes de flux de données doivent respecter des contraintes d'espace et de temps. Les données arrivant dans les flux contiennent souvent du bruit et des valeurs aberrantes. Ainsi, le clustering de flux de données doit détecter, distinguer et filtrer ces données avant la tâche de clustering.

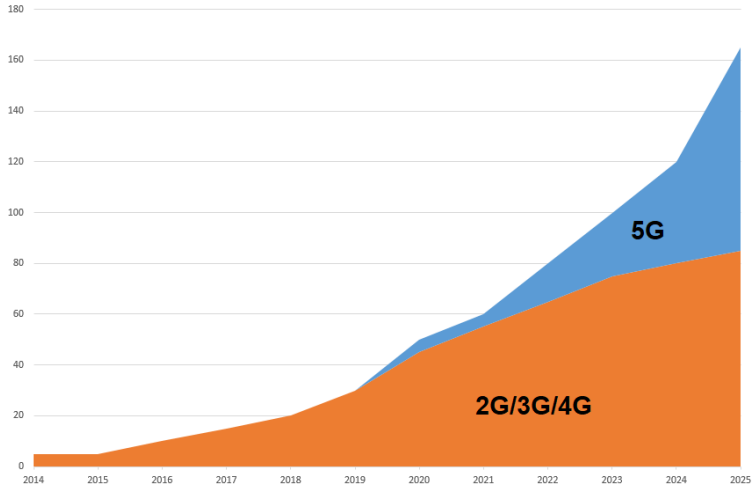


Figure 2.1 – *Trafic mondial de données mobiles (EB par mois)*

2.2 Challenges

Au CERN, l'Organisation européenne pour la recherche nucléaire, des expériences génèrent un pétaoctet entier (10^6 GB) de données chaque seconde, alors que des particules sont tirées autour du Grand collisionneur de hadrons[1]. " *Nous ne stockons pas toutes les données car cela serait peu pratique. Au lieu de cela, à partir des collisions que nous exécutons, nous ne conservons que quelques éléments intéressants, les événements rares qui se produisent, que nos filtres repèrent et envoient sur le réseau*"[2]. En d'autres termes, le CERN stocke 25PB de données chaque année ; cette quantité de données doit être analysée pour trouver des modèles qui peuvent aider à comprendre la structure et la composition de l'univers. Bien que la capacité de la mémoire ait quadruplé tous les 3 ans depuis sa création, les données croissent plus rapidement que toute capacité de mémoire dont nous disposons (2.1) et cela pose de nombreux problèmes, nous en présentons quelques-uns dans ce qui suit :

- Vitesse des données : Le flux de données évolue et arrive dans un flux non limité. Un flux de données est une séquence de données continues qui impose une restriction de passage unique. L'accès aléatoire aux données n'est pas possible.
- Flux de données à haute dimension: les progrès récents en matière d'acquisition de données n'impliquent pas seulement que les ob-

[1]https://home.cern/science/computing
[2]https://www.computing.co.uk/?v3

jets puissent être décrits dans des espaces à haute dimension. Cela conduit les techniques traditionnelles de clustering à être moins performantes lorsqu'elles traitent des ensembles de données à haute dimension.

- Temps de traitement des données : Le flux de données évolue à grande vitesse. Le flux de données entrant doit être traité à la volée.
- Restrictions de mémoire : Comme mentionné ci-dessus, les données augmentent plus vite que notre capacité à les stocker. Le stockage de ces grandes quantités de données n'est pas possible.
- Ensembles de données multi-vues : Les données peuvent être collectées à partir de plusieurs sources ou de plusieurs facettes. Dans un tel contexte, chaque point est associé à des informations beaucoup plus riches.

2.3 Contributions

Nous avons présenté différentes techniques basées sur le subspace clustering, le Multi-Objective clustering combiné avec des techniques d'analyse de flux pour répondre aux problèmes mentionnés précédemment. Nos contributions sont résumées ci-dessous :

- Nous avons présenté un état de l'art étendu sur le clustering de données statiques et de données de flux. Nous avons fourni des comparaisons entre les catégories de clustering et les algorithmes de clustering. Nous avons également présenté les techniques et les concepts de clustering de données classiques et de données de flux, ainsi que les principaux jeux de données, frameworks, ressources open-source, etc.
- Nous avons présenté une méthode de subspace clustering avec deux modèles de pondération des attributs et des blocs (global et local), une méthode efficace pour le subspace clustering d'un flux de données évolutif de manière en ligne. Nous avons utilisé les poids obtenus comme scores pour mener d'autres expériences. La méthode de subspace clustering avec le modèle de pondération globale a été utilisée comme méthode de réduction de la dimensionnalité. Nous avons démontré l'impact de l'ordre des points de données et du chevauchement des fenêtres sur la qualité du regroupement.
- Nous avons présenté une nouvelle méthode de clustering de flux de données basée sur un algorithme multi-objectif appelé MOC-Stream qui utilise deux fonctions objectives pour trouver des clusters de forme arbitraire et améliorer la qualité du clustering. MOC-Stream utilise un processus en deux phases : 1) phase en ligne : création de plusieurs solutions de clustering basées sur différents algorithmes et opérateurs génétiques 2) phase hors ligne : construction d'une partition optimale à partir des clusters découverts. Nous avons appliqué notre méthode sur de grands ensembles de données de flux et l'avons comparée à un autre algorithme de clustering de flux.

- Nous avons présenté une amélioration de la méthode précédente. La nouvelle méthode optimise le temps de calcul en utilisant les temps morts pour améliorer la solution. Elle optimise également l'allocation de mémoire en introduisant une nouvelle approche d'agrégation d'arbres pour l'algorithme Ant-Tree afin de ne stocker qu'un synopsis des données au lieu de l'ensemble des données.
- Enfin, nous avons présenté une nouvelle méthode d'extraction et de regroupement de patchs pertinents à partir d'images histopathologiques. Cette méthode combine le subspace clustering e et des techniques multi-objectifs pour surmonter le problème des taches non pertinentes. Notre approche vise à sélectionner les taches pertinentes pour la classification au lieu d'utiliser l'image entière ou toutes les régions sélectionnées en utilisant la méthode de la fenêtre glissante. Nous avons présenté deux versions de la méthode. La première version traite des ensembles de données numériques, et ses performances ont été testées sur des ensembles de données synthétiques. La seconde version traite de jeux de données RVB que nous avons adaptés pour être compatibles avec le clustering. Cette version nous permet d'extraire les patchs pertinents tout en les regroupant dans des clusters similaires.

2.4 Plan de la Thèse

Le présent travail porte sur la modélisation de données à haute dimension dans un cadre de flux de données, en utilisant le Subspace Clustering pour découvrir des clusters intégrés dans différents sous-espaces. Nous avons également utilisé le cadre multi-objectif pour faire face aux variations des caractéristiques des données. Les sections suivantes sont organisées comme suit :

1. en **Chapitre 3** : nous présentons les principales catégories de clustering et les comparons entre elles. Nous donnons également un aperçu du principal algorithme de chaque catégorie et une autre comparaison entre ces algorithmes. Nous introduisons quelques techniques et concepts de clustering et présentons quelques bibliothèques de clustering open source.
2. en **Chapitre 4** nous abordons le clustering des données de flux et ses principales propriétés et techniques. Nous donnons un aperçu des méthodes de clustering de données de flux et fournissons une comparaison entre ces méthodes. Enfin, nous présentons des ensembles de données, des cadres, des référentiels et des défis ouverts dans le domaine du clustering de données de flux.
3. en **Chapitre 5** nous présentons notre première approche, S2G-Stream avec deux modèles de pondération des caractéristiques et des blocs (Global et Local), une méthode efficace pour le clustering de sous-espaces de flux de données évolutifs de manière en ligne.
4. en **Chapitre 6**, notre deuxième approche est présentée avec deux variantes MOC et IMOC. Ces approches utilisent deux fonctions ob-

jectives pour trouver des clusters de forme arbitraire et améliorer la qualité du clustering.

5. dans **Chapitre 7**, nous abordons notre troisième approche SMO-HPS. Cette méthode combine le clustering du sous-espace et des techniques multi-objectifs pour extraire les patchs pertinents des images histopathologiques et les regrouper.

Part I

State of the Art on Clustering and Clustering Data Stream

Clustering Algorithms and Techniques 3

In this chapter we introduce the main clustering categories and compare between them. We also give an overview of the main algorithm of each category and present the merits and limitations of each one of them.

Introduction

Data clustering is a data analysis method that mines essential information from the dataset by grouping data into several groups called Clusters. In clustering, similar data points are grouped into the same cluster, while non-similar data points are put into different clusters. There are two main objectives in data clustering. The first objective is minimizing the dissimilarity within the cluster. In centroid-based clustering, this similarity is measured based on the distance between each data point and its cluster's center. It is usually calculated using the sum of squared error within the cluster. On the other hand, the dissimilarity between two clusters is calculated using the sum of squared error between clusters. According to [Xu and Wunsch, 2005], the standard process of clustering can be divided into the following several steps: (1) Feature extraction or selection: extract and select the most representative features from the original data set; (2) Clustering algorithm design: choose the right clustering algorithm for this problem; (3) Result evaluation: evaluate and validate the clustering results; (4) Result explanation: give a practical explanation for the clustering results; Figure 3.1 illustrates the clustering process.

3.1 Distances and Similarity Measures

Similarity and dissimilarity measures are crucial when constructing clustering algorithms. The similarity represents the within-cluster distances, while the dissimilarity is usually the distance between clusters. The goal of any clustering is to minimize the dissimilarity and maximize the similarity. We summarize the most used distances for numerical and for categorical data [Xu and Tian, 2015] in the following.

Minkowski distance

The Minkowski distance is a similarity measure for numerical data. It can be considered as a generalization of both the Euclidean distance and the

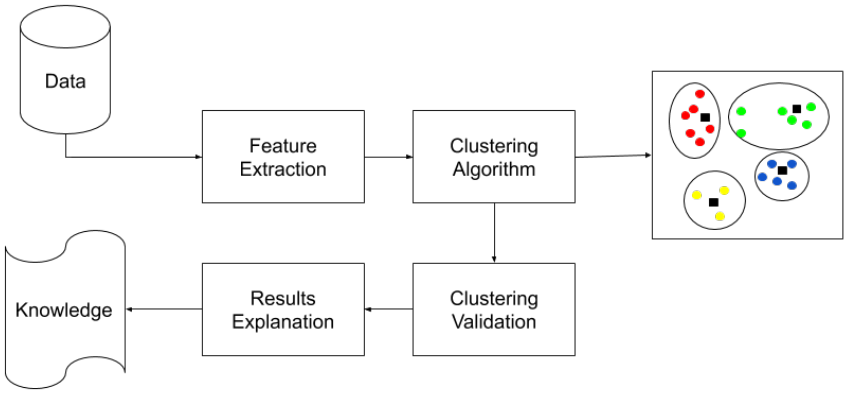


Figure 3.1 – *Clustering main steps*

Manhattan distance. The Minkowski distance is calculated as follows:

$$\left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^n\right)^{1/n}$$

When $n = 1$ the distance is equal to the Manhattan distance, when $n = 2$ it is equal to the Euclidean distance. Finally, when $n = \infty$

Cosine distance

The Cosine distance is a similarity measure used for Categorical data. It is mosly used in text mining. The Cosine distance is calculated as follows:

$$Cos(\alpha) = \frac{x_i.x_j}{\|x_i\| \, \|x_j\|}$$

Pearson correlation distance

The Pearson correlation distance is a similarity measure based on the linear correlation for numerical data. It is calculated as follows:

$$1 - \frac{cov(x_i, x_j)}{\sigma(x_i)\sigma(x_j))},$$

where cov is the covariance and σ is the standard deviation.

Mahalanobis distance

The Mahalanobis distance is a similarity measure for numerical data. It is calculated as follows:

$$\sqrt{(x_i - x_j)S^{-1}(x_i - x_j))},$$

where S is the covariance matrix inside a cluster.

Jaccard distance

The Jaccard distance is a dissimilarity measure for Categorical data. It measures similarity between finite sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets. It is calculated as follows:

$$J(X,Y) = \frac{X \cap Y}{X \cup Y}$$

Hamming distance

The Hamming distance is a dissimilarity measure for Categorical data. It represents the minimum number
of substitutions needed to
change one data point into another. It is calculated as follows:

$$d(x_i, x_j) = \sum_{k=1}^{n} (x_{ik} \bigoplus x_{jk})$$

3.2 Validity Measures

Quality or Validation Measures are very important to validate a clustering solution and measure its quality. Several measures are presented in the literature. These measures are grouped into two main families, internal and external measures. We present a definition for those two classes and some well-used examples in the next sections.

3.2.1 Internal Validity Measures

Based on the information intrinsic to the data alone. It does not require a-priori knowledge of the data. It is suitable for unsupervised learning as we don't have labeled datasets. We list in the following some of these validity measures.

Silhouette Index SI

The Silhouette index [Rousseeuw, 1987] compute the compactness and the separateness of clusters. For a data point x_i assigned to cluster C_i, the Silhouette index is calculated as follow:

$$SI(i) = \frac{(b(i) - a(i))}{Max(b(i) - a(i))} \tag{3.1}$$

Where $a(i)$ is the average distance between x_i and all the data points assigned to cluster C_i. $b(i)$ is the minimum average between x_i and the data points assigned to cluster C_j where $j = 1, .., K; j \neq i$.

Mean Square Error MSE

MSE calculates the compactness of a cluster, its calculated as follows:

$$MSE = \frac{1}{N} \sum_{k=1}^{K} \sum_{x_i \in C_i} d(x_i, C_i) \tag{3.2}$$

Where K is the number of clusters, $d(x_i, C_i)$ is the distance between data point x_i and cluster C_i.

Dunn Index

Dunn index [Dunn, 1973] is the ratio between the maximum distance between two points clustered together and the minimum distance between two points clustered separately. Its calculated as follows:

$$Dunn_i = min(\frac{d(C_i, C_j)}{max(d(x_i, C_j))}) \tag{3.3}$$

Where $d(C_i, C_j)$ is the distance between cluster C_i and cluster C_j, $d(x_i, C_j)$ is the distance between data point x_i and cluster C_j and $j = 1, .., K; j \neq i$. K is the number of clusters.

Davies Bouldin

Davies Bouldin index [Davies and Bouldin, 1979] helps identify sets of clusters that are compact and well separated. The Davies-Bouldin index is calculated as:

$$DBI = \frac{1}{K}\sum_{i=1}^{K} max_{i,j=1,..,K;j\neq i}\frac{d(x_i, C_i) + d(x_j, C_j)}{d(C_i, C_j)} \tag{3.4}$$

Where $d(x_i, C_i)$ is the distance between the data point x_i and its cluster C_i and K is the number of clusters.

BIC Index

The Bayesian information criterion BIC [Raftery, 1986] is an index used to avoid overfitting, it is calculated as follows:

$$BIC = -ln(L) + vln(n) \tag{3.5}$$

Where n is the number of data points, L is the likelihood of the parameters to generate the data in the model, and v is the number of free parameters in the Gaussian model.

3.2.2 External Validity Measures

Based on previous knowledge about data. It uses a labeled dataset to validate the predicted labels. This type of validity measure is mostly used for supervised learning.
The comparison between two clustering solutions C and $C^{'}$ can be made by a contingency matrix, where n_{ks} represents the number of points assigned to both clusters k and l of partitions C and $C^{'}$:

	Y_1	Y_2	...	Y_s	Sums
X_1	n_{11}	n_{12}	...	n_{1s}	a_1
X_2	n_{21}	n_{22}	...	n_{2s}	a_2
...	...	...	...	...	...
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

Table 3.1 – *Contingency matrix between two partitions C and $C^{'}$ of r and s clusters respectively.*

The contingency matrix is used to define the following quality measures:

Normalized Mutual Information NMI

NMI [Strehl and Ghosh, 2002] provides a measure that is independent of the number of clusters as compared to purity. It reaches its maximum value of 1 only when the two sets of labels have a perfect one-to-one correspondence. The NMI of a clustering solution C is calculated as follows :

$$NMI(Y,C) = \frac{2 \times I(Y;C)}{H(Y) + H(C)} \tag{3.6}$$

Where Y are true labels and C are labels predicted by the algorithm. $I(Y;C) = H(Y) - H(Y|C)$ and $H(C)$ is the entropy of the partition calculated as follow.

$$\sum_{k=1}^{K} \frac{n_k}{N} log(n_k N) \tag{3.7}$$

Where n_k is the number of points assigned to the partition k.

Adjusted RAND index ARI

ARI index [Hubert and Arabie, 1985] is a measure of agreement between two partitions: one given by the clustering process and the other defined by external criteria. The Adjusted RAND index is calculated as follows:

$$ARAND = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}} \tag{3.8}$$

Precision

The precision index indicates the probability that two data points are clustered together in partition $C^{'}$ if they are clustered together in partition C :

$$precision(C,C^{'}) = \frac{n_{11}}{n_{11} + n_{01}} \tag{3.9}$$

Recall

The recall index indicates the probability that two data points are clustered together in partition $C^{'}$ if they are not clustered together in partition C :

$$recall(C,C^{'}) = \frac{n_{11}}{n_{11} + n_{10}} \tag{3.10}$$

F-measure

F-measure is the harmonic mean of indices precision and recall. It is calculated as follows:

$$\mathcal{F}(C, C^{'}) = \frac{precision(C, C^{'}) \times recall(C, C^{'})}{precision(C, C^{'}) + recall(C, C^{'})} \tag{3.11}$$

Purity

The purity of a partition is the quantity of the the consistency of one partition with respect to another. It is calculated as follows:

$$purity(C, C^{'}) = \frac{1}{N} \sum_{k=1}^{K} argmax_{C_l}(n_{kl}) \tag{3.12}$$

3.3 Classic Clustering Algorithms

We can classify clustering into many categories based on how the clusters are retrieved. We list some of the main categories in the following sections:

3.3.1 Partitioning Clustering

A partitioning method divides the dataset into several partitions. Each partition is considered as a cluster and the division criterion is the dissimilarity measure, e.g., Euclidean distance, K-Means [MacQueen, 1967] is the most famous partitioning clustering algorithm. The main idea of K-means is to represent a cluster by its center. The initial K, which represents the number of clusters, is randomly initialized. K-means iteratively assigns each point to the cluster with the closest center. Then it recalculate the centers, the iterative process will be continued until some criteria for convergence is met or after a number of iterations t. The K-means algorithm is presented in Algorithm (1)

Algorithm 1 K-means

input : K and a set of points N
output : K Clusters of points and their centers
while *convergence criterion not met* **do**
- (re)assign each point to its closest center based on the euclidean distance ;
- Compute new centers as the mean of the new points assigned to the cluster ;

K-medoids [Kaufman and Rousseeuw, 1990] is an improvement of K-means to deal with discrete data, which takes the data point, most near the center of data points, as the centroid of the cluster. other algorithms for K-medoids clustering have been developed. [Kaufman and Rousseeuw, 1990] proposed an algorithm called CLARA, which applies the K-medoids algorithm to sampled objects instead of all objects. The performance of CLARA drops rapidly below an acceptable level with increasing number of clusters. [Lucasius et al., 1993] proposed a new approach of K-medoid

clustering using a genetic algorithm, whose performance is reported as better than CLARA but computational burden increases as the number of clusters increases.

CLARANS (Clustering Large Applications based upon RANdomized Search) [Ng and Han, 2002]: It presents a trade-off between the cost and the effectiveness of using samples to obtain clustering. The clustering process can be given as looking for a graph where every node is a potential solution, a set of k randomly selected medoids. For each medoid x, CLARANS tries to find another object y that can replace x while improving a criterion. The process is repeated till the final result is obtained.

3.3.2 Hierarchical Clustering

This method creates a hierarchical relationship among data points in a tree-like structure called a dendrogram, and there are two types of hierarchical methods :

- Agglomerative Methods(AGNES) this method uses a dissimilarity measure to group the nodes with low dissimilarity two by two, this can lead to all nodes be in the same group.
- Divisive Methods (DIANA) is the inverse of AGNES, begins with N points and at each iteration it chooses a segment to divide.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [Zhang et al., 1996] is a hierarchical clustering algorithm; it was originally designed to handle classic data. However, it has also been used to clustering data streams due to its suitability for use with high dimensional data. It constructs a tree of clustering features called CF-tree. The CF-Tree has two user-defined parameters: branching factor B and the diameter T; B defines the maximum entries that a node can contain; T is a threshold that a node must not violate when absorbing a new entry. Each non-leaf node contains at most B entries of the form $[CF_i, child_i]$ where $i = 1, ..., B$. A leaf node represents a cluster made up of all the sub-clusters represented by its entries; it contains at most L entries of the form $[CF_i]$ where $i = 1, ..., L$. In addition, every leaf node has two pointers, prev and next, which are used to chain all leaf nodes together. Figure (3.2) shows an example of a CF-tree.

Each entry in the CF-tree represents a cluster of objects and is characterized by a 3-tuple: (N, LS, SS), where N is the number of objects in the cluster and LS, SS are defined in the following:

$$LS = \sum_{P_i \in N} P_i$$

$$SS = \sum_{P_i \in N} \mid P_i \mid^2$$

when a new data point comes, this statistics are updated as follow:

$$N_i = N_i + 1$$

$$LS_i = LS_i + x$$

$$SS_i = SS_i + x^2$$

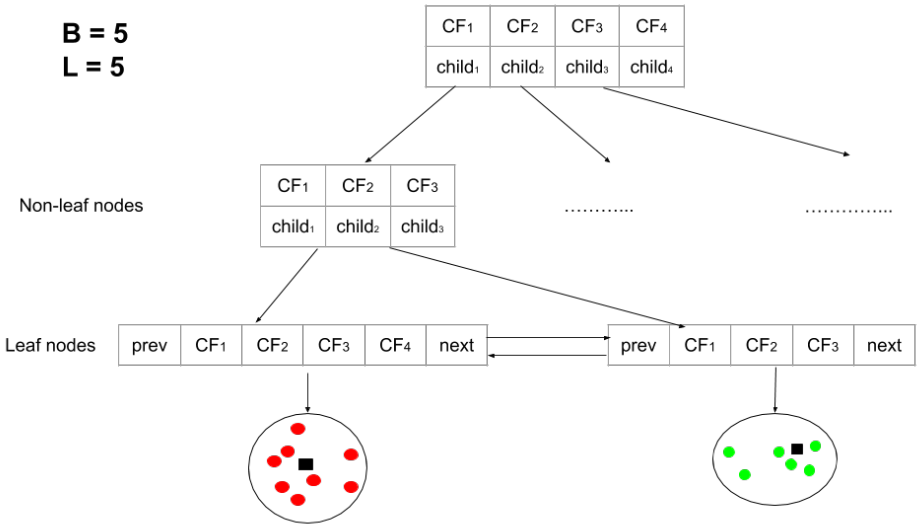


Figure 3.2 – *BIRCH's CF-tree example*

Additivity theorem allows us to merge sub-clusters incrementally and consistently .

$$CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2) \tag{3.13}$$

To insert a new data point, an appropriate leaf needs to be identified by starting from the top of the tree and recursively descend by choosing the closest child node according to a proper metric. After inserting an entry into a leaf, the CF information for each non-leaf entry on the leaf path is updated. When a node is split, a new non-leaf entry is inserted into the parent node and pointed to the newly formed leaf. According to B, the parent doesn't have enough room, therefore it will be split as well, and so on up to the root.

CURE [Guha et al., 1998] is a hierarchical clustering algorithm. It uses random sampling (4.2) to produce samples that will be clustered separately; those clusters are integrated into the final solution. This algorithm is suitable for large-scale clustering. ROCK [Guha et al., 2000] is an improvement of CURE to deal with categorical data, which takes into consideration the effect on the similarity from the data around the cluster.

3.3.3 Density-Based Clustering

Density-based algorithms use density functions as similarity measures; the dense areas are considered clusters, separated by low-density regions (outliers). This kind of algorithm presents several advantages, such as the ability to treat noise or also the ability to detect clusters of arbitrary shapes. DBSCAN (density-based spatial clustering of applications with noise) [Ester et al., 1996] is the most used density-based clustering algorithm. It considers the points with the most neighbors as centers of the clusters (the regions with high density are considered clusters). At the same time, points with fewer neighbors are considered noise. The DBSCAN algorithm introduces two concepts of Clustering: Reachability and Connectivity.

- Reachability: a point is reachable from another if the distance between them is inferior to a threshold ϵ.
- Connectivity: if two points p and q are connected they belong to the same cluster. If points p and r are connected and r and q are connected, then p and q are also connected (transitivity).

To apply these concepts, DBSCAN introduces two parameters: *MinPts* and ϵ.

- *MinPts*: the minimum of points that a region should have to be considered dense.
- Threshold ϵ: a threshold to determine if a point belongs to another point's neighborhood.

Based on the concepts described above, DBSCAN presents three types of points:

- Core point: it has at least *MinPts* points within a distance of ϵ.
- Border point: it is not a core point, but it belongs to at least one cluster. That means that it lies within a distance ϵ from a core point.
- Noise point: it's a point that is not a core point nor a border point.

Figure (3.3) illustrates DBSCAN's point types.
The DBSCAN algorithm proceed by taking point randomly from the

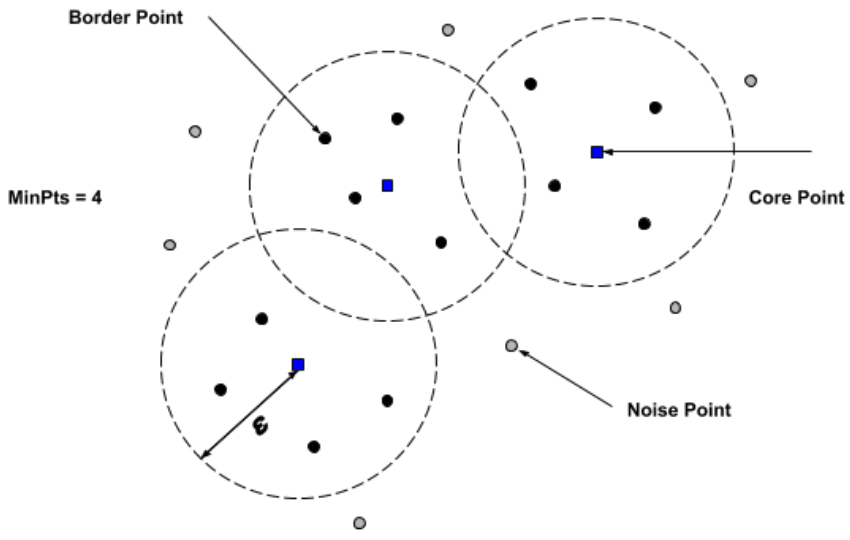


Figure 3.3 – *DBSCAN point types*

dataset until all points are picked. Then for each point it determines if it's a Core, Border or Noise point based on the parameters ϵ and *MinPts*. The core point are considered centers of each cluster. The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point. Algorithm (2) present the DBSCAN algorithm. The

main problems of DBSCAN are its sensitivity to varying density. It is also not suitable with high dimensional data.

Algorithm 2 DBSCAN

input : ϵ, *MinPts* and a set of points N
output : Clusters of points and their centers
while *there is a non processed point in the dataset* **do**
- Randomly select a non processed point p ;
- Compute the distance between this point and all the other points in the dataset and retrieve the points within a distance ϵ ;
- If p is a Core point a cluster is formed ;
- If p is a Border point it will be marked as member of a cluster ;
- If p is not a Border nor a Core point, it is considered Noise point ;

OPTICS [Ankerst et al., 1999] is an improvement of DBSCAN, and it overcomes the DBSCAN's shortcoming that being sensitive to varying density by introducing two parameters: the Core Distance: which is the minimum value of radius required to classify a given point as a core point. It introduces the Reachability which is the distance between a point p and q, which is the maximum of the Core Distance of p and the Euclidean Distance(or some other distance metric) between p and q. Note that The Reachability Distance is not defined if q is not a Core point.

3.3.4 Grid-Based Clustering

Grid-Based Clustering Methods change the data space into several cells that form a grid structure; this algorithm is designed for spatial data. The data points are mapped into the cells. The cells are then merged based on their density. The main algorithms of this kind of clustering are STING [Wang et al., 1997] and CLIQUE [Agrawal et al., 1998a]. STING (Statistical Information Grid) is used for parallel processing. It divides the data space into many rectangular units by constructing the hierarchical structure. The data within different structure levels are clustered, respectively. The method determines a layer to begin with. Each cell of this layer calculates the confidence interval (or estimated range) of the probability that this cell is relevant to the query. From the interval calculated above, it labels the cell as relevant or not relevant. If this is the bottom layer, then it ends the process. Otherwise, it goes down the hierarchy structure by one level and repeats the steps. Figure (3.4) illustrates the different layers of the grid in the algorithm STING.

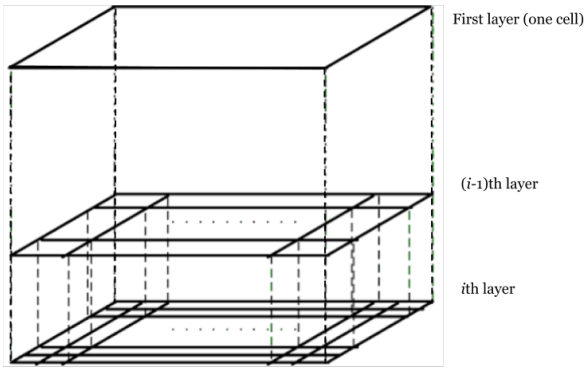


Figure 3.4 – *GRID layers in STING algorithm*

WaveCluster [Sheikholeslami et al., 1998] is a grid clustering algorithm that applies wavelet transform on the spatial data feature space. This helps detect arbitrary shape clusters at different scales. WaveCluster is insensitive to the order of input data to be processed. It is also not affected by the outliers and can handle them properly.

3.3.5 Model-Based Clustering

The basic idea is to select a particular model for each cluster and find the best fitting for that model. COBWEB [Fisher, 1987] is the most known model-based clustering algorithm; it constructs a tree-like hierarchy of data based on a category function. Each node of the tree keeps a notion and has a probabilistic description of that notion, which resumes the objects classified under the nodes. To construct the hierarchy, COBWEB sorts each point into its tree, and at each node, it considers four operations to incorporate the new example into its tree [McKusick and Thompson, 1990]:

- Merging Two Nodes: applied when the hierarchy is overly branched, and combining two classes provides a good concept to which to classify the incoming instance.
- Splitting a node: applied when the hierarchy contains a node that is too general and, therefore, less useful for classification and prediction.
- Inserting a new node: an instance is added if it fits into an existing node well. This operator integrates the instance into one of the child nodes. If this child node is not a singleton (i.e., it describes more than one instance), COBWEB updates the conditional probabilities for the node and each of the attribute values.
- Creating a node: applied when an instance has very different characteristics from any existing concept at the current level, as determined by its evaluation function. This operator places the instance in a category by itself, a sibling of the existing concept nodes.

Figure (3.5) illustrates the tree operations. Once COBWEB has constructed a tree, it can be used to return the clustering of the points at varying levels of aggregation. The main advantage of the COBWEB algorithm is the capacity to detect noise and missing values. Self Organizing Maps [Kohonen, 1998] is another well-known model-based clustering algorithm. The core idea of SOM is to map the input space of high dimension into output space of low dimension called topological map on the assumption that there exists topology in the input data. In this map, each class is represented by a neuron, which is characteristic by a referent vector(prototype); SOM uses a neighborhood function to preserve the input space's topological properties.

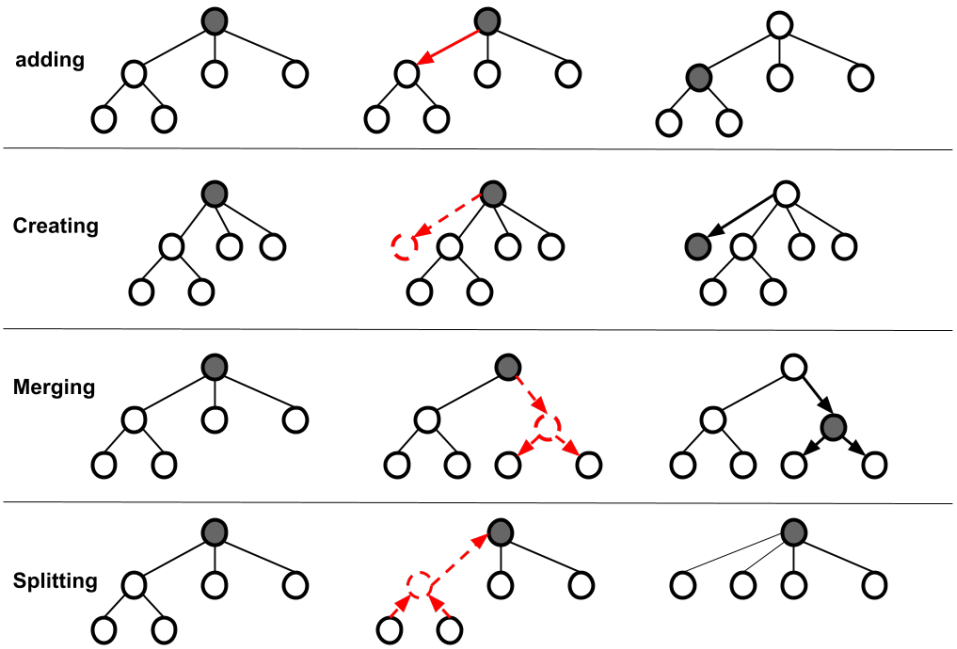


Figure 3.5 – *COBWEB's tree operations*

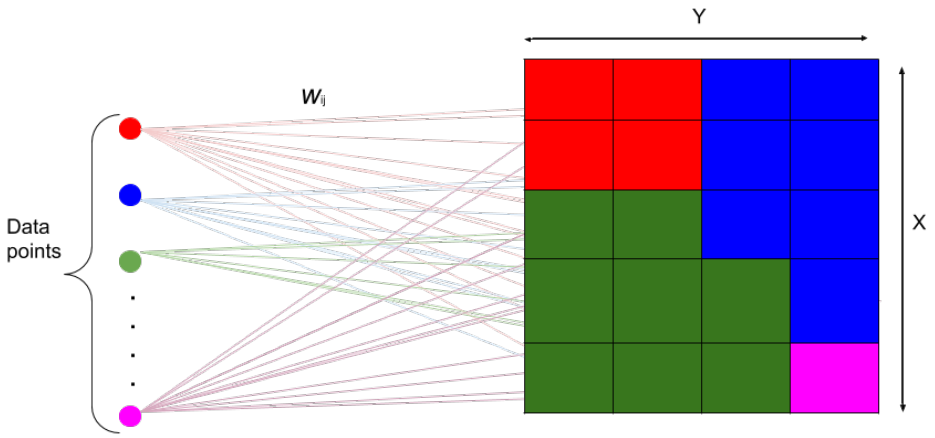


Figure 3.6 – *Example of data projection in SOM algorithm. w_{ij} represents the weight of point x_i in the cell j. X and Y are the dimensions of the map.*

Growing Neural Gas is a variant of SOM [Fritzke, 1995]. It is an incremental clustering algorithm. Given some input distribution, GNG incrementally creates a graph of nodes. Each node in the graph represents a cluster prototype and has a position in the input distribution. GNG can be used for finding topological structures that better represent the structure of the input distribution. GNG is an adaptive algorithm in the sense that if the input distribution slowly changes over time, GNG can adapt to move the nodes to cover the new distribution.

Starting with a random two nodes, the algorithm constructs a graph in which nodes are considered neighbors if an edge connects them. For each point x, an edge is inserted between the two closest nodes. The goal of GNG is to minimize the quantization error; therefore, the clusters that present high quantization errors are split into smaller clusters. To do so, GNG insert a new node between the nodes of the clusters with the highest error using the following equation:

$$w_{new} = \frac{1}{2}(w_1 + w_2) \tag{3.14}$$

Where w_{new} is the prototype of the new node. Figure (3.7) illustrates the graph evolution of GNG algorithm.

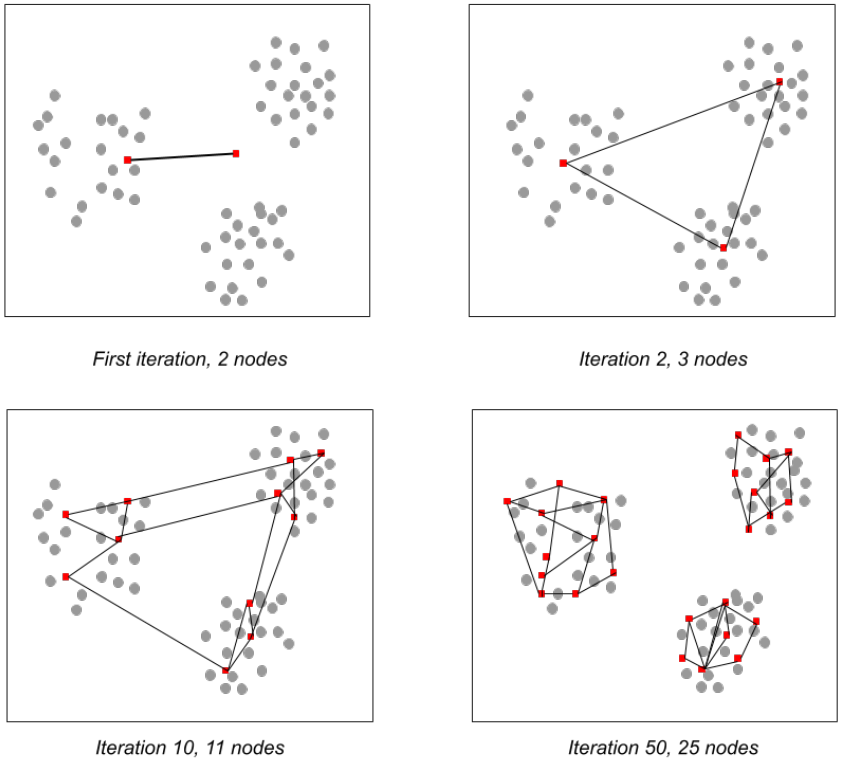


Figure 3.7 – *Example of the graph evolution in GNG algorithm.*

3.3.6 Ensemble Clustering

These approaches effectively combine the benefits of the other two types of methods and make it possible to cancel the disadvantages, and The objective is to use the opinion of several partitions in order to reach a consensus or a synthesis. These partitions can be obtained by varying clustering algorithms, initial parameters, subsets of data, etc.
Ensemble clustering methods attempt to find the consensus partition summarizing, at best, a given set of partitions. These methods are divided into two stages, generation and consensus. In the generation phase, several models are generated using different methods like different algorithms or different initialization settings. The consensus phase is a function that takes as input the N results and produces a clustering result. There are two main consensus functions: co-occurrence of objects and median partition. The first function calculates how many times an object belongs to a cluster. A consensus is obtained by voting among the objects. Each object votes for the cluster to which it must belong. Voting k-means uses different values of K to generate different models. Then it uses a vote to determine the consensus. Median partition has been proven to be an NP-hard optimization problem. Its purpose is to calculate the similarities between different models generated then to find the model that maximizes similarity with all other models.

IDEStream Khan et al. [2016a] has been applied in data streams. It uses the k-means algorithm at first to group N points in $\sqrt{N}$ micro-clusters. It estimates each incoming characteristic vector's probability of density using the degree of dispersion to detect aberrant points. Then it defines the

cluster number by applying the Charad method and using the Krzanowski criterion Krzanowski and Lai [1988], then uses a set method to aggregate the clustering of two-time windows in one, and then the same process is repeated. The goal is to find the clustering that minimizes the information variation VI between clusterings. For this purpose, the method uses a reinforcement learning equation.

SEFCM (Fuzzy C-Means Ensemble Stream) Fathzadeh and Mokhtari [2013] divides the data stream into b blocks with size b/w and then applies the set-theorem algorithm to each block to generate k clusters by choosing initial settings randomly for each partition which guarantees the diversity of the results. The resulting bk clusters will be grouped into k new clusters. Then, the method combines the k partitions to come out with an optimal partition by applying the EFCM algorithm.

UBLA Billot et al. [2008] (Unsupervised Boosting-Like Approach) is an ensemble method for clustering based on boosting algorithm.it is a four steps methods :

1. **Evaluation**: calculates the quality criterion from the membership degrees for each point, which will be weighted at the end of this phase.
2. **Grouping**: a sampling method is used to build ten samples taking into account the weights already assigned.
3. A co-association matrix is used and updated during the iterations. The first three steps are repeated.
4. Establish the final score.

3.3.7 Evolutionary Clustering

Evolutionary approaches are inspired by natural evolution. They make use of evolutionary operators (selection, crossover and mutation) and a population of solutions to obtain the globally optimal partition of the data. The clustering here can be viewed as an optimization problem that finds out the optimal centroid of the clusters. Each clustering solution is encoded using an encoding scheme (Section (3.5)), then, a random population of clustering solutions is set. The algorithm calculates the fitness value for each solution. For each iteration, the algorithm applies evolutionary operators on the solutions, and recalculate the fitness values. The process is repeated until there is a convergence of the solutions. The optimal solution is chosen as the solution with the highest fitness value.

ACOC algorithm is an Ant Colony clustering algorithm [Kao and Cheng, 2006]. It produces the final solution as a graph $d \times k$, where k is the pre-defined number of clusters, and d is the number of points. Each node represents the assignment of a point to a particular cluster. Figure (3.8) illustrates an example of graph construction in ACOC algorithm. White circles represent non-visited nodes and solid circles represent visited nodes. A string is used to represent solutions built by ants. Considering the clustering result of Figure (3.8), the corresponding solution string is (2, 3, 2, 4, 3). Each ant moves from one node to another, deposits pheromone on nodes, and constructs a solution in a stepwise way. The algorithm uses a memory list to prevent a data point from being clustered

more than once by an ant. When the memory list is full, it means that the ant completes solution construction. ACOC uses a pheromone matrix (PM) to store pheromone values. The nodes with stronger pheromone would be more attractive to ants. The assignment function is obtained by calculating the Euclidean distance between the data point to be grouped and each cluster center of some ant. Each ant carries a cluster center matrix to store its own cluster centers and updates them right after each clustering step.

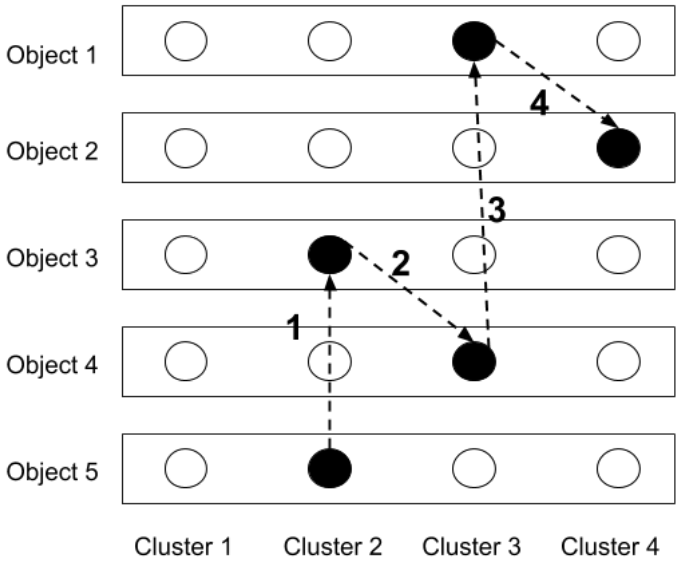


Figure 3.8 – *Graph construction in ACOC algorithm*

3.3.8 Comparison Between Clustering Methods

Table (3.2) compare between some clustering methods from each category. The merits and the limitations of each clustering category are presented in the following.

- **Density-Based Clustering**: Detect arbitrary shaped clusters, Highly efficient and Handle Noise. In the other hand, the clustering result is highly sensitive to the parameters. And the method Does not work well in multi density data.
- **Hierarchical Clustering**: Easy to handle any measure and Can determine clusters with arbitrary shape. But is presents an ambiguity of termination criteria and a high complexity.
- **Partitioning Clustering**: Easy to implement and present Low time complexity and high computing efficiency. However, the number of clusters must be predefined by user and Can not determine clusters with arbitrary shape.
- **Grid-Based Clustering**: Detect arbitrary shaped clusters, Handle noise and have a low time complexity. It is not Adaptable with high dimensional dataand the clustering result sensitive to the granularity.

- **Model-Based Clustering** : Specify the number of clusters automatically based on standard statistics. It can detect outliers. However, this category of methods have a high time complexity and the clustering result depends on the appropriate choice of the number and width of the partitions and grid cells.
- **Ensemble Clustering**: Robust and scalable, it takes advantage of the used algorithms. However, we need to find the best consensus method and the methods employed are time consuming.
- **Evolutionary Clustering**: Easy to implement and Highly efficient. This methods have a high time complexity and they are not Scalable and not suitable for high dimensional data.

Category	Typical Algorithm	Complexity	Scalability	HDD[1]	ASC[2]	NS[3]
Density	DBSCAN	O($n \times \log n$)	✓	✗	✓	✓
Partitioning	K-means	O($K \times n \times$ iterations)	✗	✗	✗	✓
Hierarchical	BIRCH	O(n)	✓	✗	✗	✓
Grid	STING	O(n)	✓	✓	✓	✗
Model	COBWEB	N/A[*]	✓	✗	✓	✓
Evolutionary	ACO-based	High	✗	✗	✗	✓

[1] High Dimensional Data
[2] Arbitrary-shaped clusters
[2] Noise sensitivity
[*] N/A: not available

Table 3.2 – *Comparison between clustering categories*

3.4 Subspace clustering

Subspace clustering discovers clusters embedded in multiple, overlapping subspaces of high dimensional data. It is an extension of feature selection, which tries to identify clusters in different subspaces of the same dataset. As a feature selection, subspace clustering needs a search method and an evaluation criterion. Also, subspace clustering must somehow restrict the scope of the evaluation criterion to consider different subspaces for each distinct cluster. We present a state of the art of the subspace clustering techniques and algorithms in the next sections.

3.4.1 Search Methods

Subspace clustering consists of finding each class in a subspace composed of relevant features, and a feature may be suitable for one or more clusters. More sophisticated heuristics that can be grouped into two categories are then developed to optimally determine the subspaces associated with the classes of the classification. Based on the way subspaces are determined, subspace clustering methods are classified into two main categories: Hard Subspace Clustering (HSC) and Soft Subspace Clustering (SSC). SSC algorithms perform clustering in high dimensional spaces by assigning a weight to each feature to measure the contribution of individual features in the formation of a particular cluster [Deng et al., 2016]. In HSC, all

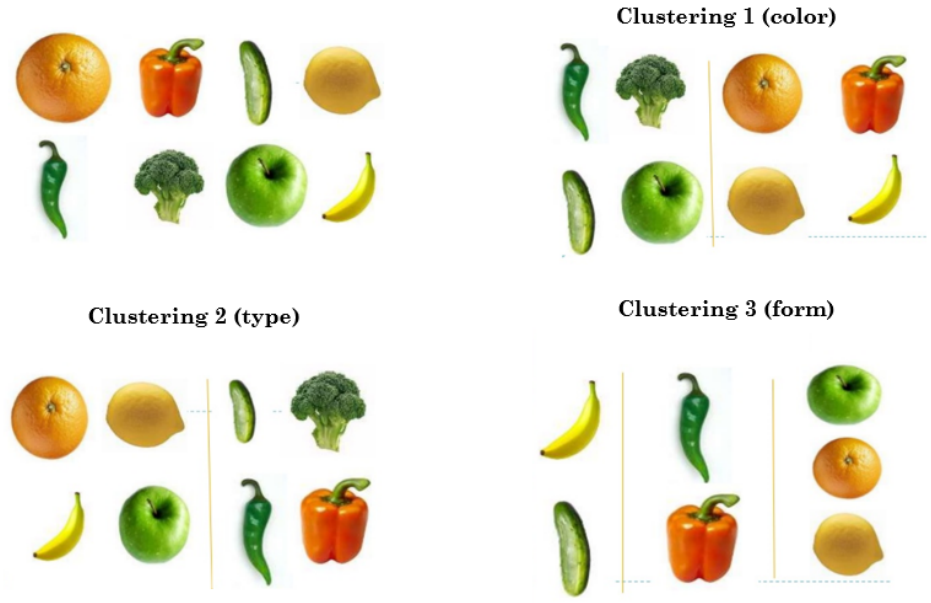


Figure 3.9 – *Subspace clustering illustration*

features contribute equally to the clustering process. Figure (3.10) shows a classification of subspace clustering algorithms.

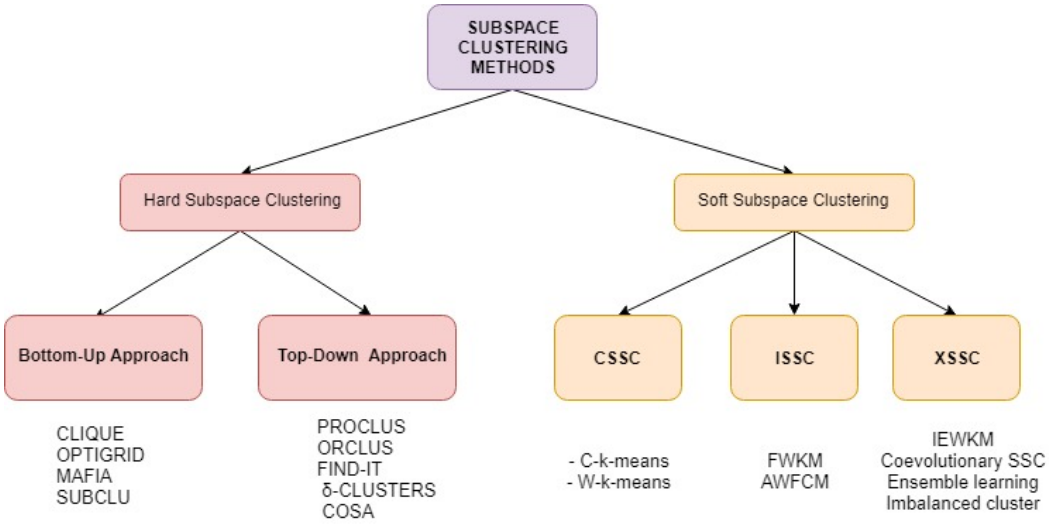


Figure 3.10 – *Hierarchy of Subspace Clustering Algorithms.*

Hard Subspace Clustering HSC

HSC algorithms are divided into bottom-up and top-down search methods.

Top-down approaches determine the first clustering using all the features. A weight associated with each feature is then used in a new phase of an iterative process to reassign the observations to the classes. The main difficulty in this category is the determination of the number of clusters and the number of features forming the subspace associated with a cluster.

Bottom-Up approaches use clustering methods based on a mesh of the observation space by defining a histogram for each dimension. Then,

intervals with a density of observations above a threshold set a priori to denote clusters for each feature. They are starting from 1-dimensional clusters, which are combined iteratively to form the clusters in the higher dimensional subspaces [Parsons et al., 2004]. However, these algorithms can find arbitrary-shaped subspace clusters but fail to scale with the dimensions.

Soft Subspace Clustering SSC

SSC (Soft Subspace Clustering) algorithms assign a weight to each dimension in a clustering process viewed as the degree of contribution of that dimension in that cluster. After the clustering process, the weights identify the subspaces of different clusters. The purpose of these algorithms is to select important features from the whole dataset. From this perspective, soft subspace clustering can be viewed as multiple feature weighting clustering. SSC methods can be classified into three categories [Deng et al., 2016]: CSSC, ISSC and XSSC.

Firstly ***CSSC* (Conventional Subspace Clustering)** uses a feature weighting process in a two steps clustering process. First, it uses some weighting strategies to find subspaces. Then clustering is performed on the subspace that was obtained (separated feature weighting). Clustering can also be obtained by performing the two processes simultaneously (coupled feature weighting) [Deng et al., 2016].

Secondly ***ISSC* (Independent Subspace Clustering)** method: each cluster has its weight vector to form its subspace. This kind of clustering uses many weighting methods such as fuzzy and entropy weighting.

Finally, ***XSSC* (Extended Subspace Clustering)** has been proposed to enhance the performance of CSSC and ISSC, employing many strategies to improve the clustering process.

3.4.2 Weighting Models

In Soft Subspace clustering methods, the subspace is determined through weights as a score of the relevance of the subspace. Therefore, the choice of a good weighting model is crucial to the clustering process and subspace determination. We present some of the well-known weighting models in the following.

· **Separated feature weighting** In this model, the clustering process is separated from the subspace identification process. The subspaces are retrieved first, and then, the clustering algorithm is applied to these subspaces to get the final clustering.

· **Coupled feature weighting** On the contrary of the previous model, in coupled feature weighting, both clustering and subspace processes are used simultaneously to obtain the final clustering.

· **Fuzzy weighting** Each data object has a fuzzy membership, which can be seen as the probabilities that a data object (or feature) belongs to each one of the existing clusters.

· **Entropy weighting** Entropy is a measure of uncertainty of a random variable. The method is motivated by the fact that a subspace with clusters typically has lower entropy than a subspace without clusters.

· **Coevolutionary feature weighting** Several evolutionary methods are employed to produce different populations. These populations cooperate to elect the best solution: the better the quality of the global result, the better the individual evaluation. They search for the partition built from local solutions that minimize a cost function [Gançarski et al., 2008a].

3.4.3 Subspace Clustering Algorithms

Hard Subspace Clustering HSC

HSC algorithms are divided into bottom-up and top-down search methods [Friedman and Meulman, 2004]. Top-down approaches determine the first clustering using all the features. A weight associated with each feature is then used in a new phase of an iterative process to reassign the observations to the classes. The main difficulty in this category is the determination of the number of clusters and the number of features forming the subspace associated with a cluster. PROCLUS is a top-down approach [Aggarwal et al., 1999], which uses three phases (initialization, iteration, and refinement). In the first step, a greedy algorithm selects a set of potential medoids. Then a set of dimensions is computed corresponding to each medoid so that points assigned to the best medoid form a cluster in the subspace determined by those dimensions. In the end, new dimensions of each medoid are computed based on the obtained clusters. Then, points are reassigned to medoids, and outliers are removed.

Bottom-Up approaches use clustering methods based on a mesh of the observation space by defining a histogram for each dimension. Then, intervals with a density of observations above a threshold set a priori to denote clusters for each feature. They are starting from 1-dimensional clusters, which are combined iteratively to form the clusters in the higher dimensional subspaces [Parsons et al., 2004]. These algorithms can find arbitrary-shaped subspace clusters but fail to scale with the dimensions.

CLIQUE [Agrawal et al., 1998b] is a density-based and grid-based subspace clustering algorithm. It uses a two-phase model, which makes it suitable for stream clustering. In the online phase, data points are projected into a grid. A density-based clustering is used in the offline phase to produce the final clustering from the dense cells. Figure (3.11) illustrates the different clusters detected in other subspace with the CLIQUE algorithm.

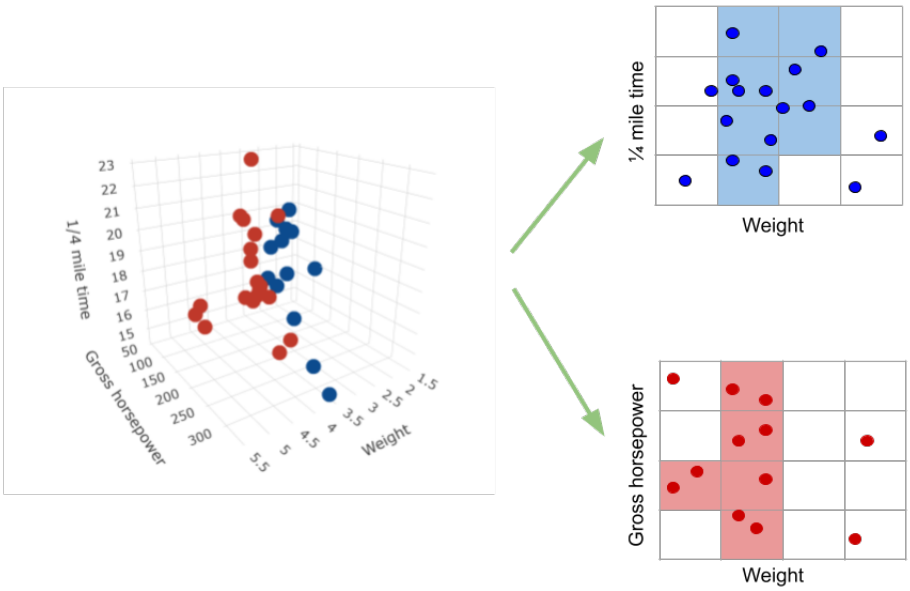


Figure 3.11 – *Subspaces in CLIQUE algorithm*

CLIQUE algorithm proceeds in 3 major steps:

1. Identify subspaces that contain clusters :
   - Partition the data space and find the number of points that lie inside each cell of the partition
   - Identify the subspaces that contain clusters using the Apriori principle
2. Identify clusters
   - Determine dense units in all subspaces of interests
   - Determine connected dense units in all subspaces of interests
3. Generate minimal descriptions for the clusters
   - Determine maximal regions that cover a cluster of connected dense units for each cluster
   - Determine minimal cover for each cluster

CLIQUE automatically finds subspaces and high-density clusters in them. However, As in all grid-based clustering approaches, the quality of the results crucially depends on the appropriate choice of the partitions and grid cells' number and width.

Soft Subspace Clustering SSC

C-k-means (Convex k-means) is a CSSC method that uses a separated feature weighting [Modha and Spangler, 2003]. It involves two separate processes: subspace identification and clustering in subspace. It begins by assigning a set of weights to each data, which is not practical for high dimensional data. Another method named W-k-means uses a coupled feature weighting [Huang et al., 2005], which means that the weights are

updated adaptively in the clustering process.
AWFCM [Keller and Klawonn, 2000] is an ISSC; it uses fuzzy weighting and takes into account all features in the clustering task. The feature selection is carried out during the learning phase providing information about the influence of selected features.
In [Chan et al., 2004], authors present an XSSC method called IEWKM (Improved Entropy Weighting K-means) by developing a new procedure to generate the weight for each attribute from each cluster within the framework of the k-means-type algorithm. Coevolutionary SSC [Gançarski et al., 2008b] introduces a weighting system based on a coevolutionary learning technique. The coevolutionary algorithm extends the evolutionary methods to deal with complex problems. It uses several populations; each one of them is evolved in an environment that depends on the other population. The population is combined with its collaborators to form a complete solution, and an objective function is evaluated. Ensemble learning approaches [Domeniconi and Al-Razgan, 2009] combine the benefits of the other two types of methods. Its goal is to use the opinion of several partitions to reach a consensus or a synthesis. These partitions can be obtained by varying clustering algorithms, initial parameters, subsets of data, etc.

Table (3.3) presents a comparison between some subspace clustering methods.

Table 3.3 – *Comparison Between Subspace Clustering Algorithms*

Method	Category	Search method	Weighting method	Overlap of dimensions	Data type
CLIQUE	Hard	Bottom-Up	-	✓	Mixed
MAFIA	Hard	Bottom-Up	-	✓	Numerical
PROCLUS	Hard	Top-Down	-	✗	Numerical
FINDIT	Hard	Top-Down	Dimension voting	✗	Numerical
C-k-means	Soft	CSSC	Separated FW[1]	✓	Mixed
W-k-means	Soft	CSSC	Coupled FW[1]	N/A	Mixed
FWKM	Soft	ISSC	Fuzzy weighting	✓	Mixed
AWFCM	Soft	ISSC	Fuzzy weighting	N/A	Numerical
IEWKM	Soft	XSSC	Entropy weighting	N/A	Mixed
CSSC	Soft	XSSC	Coevolutionnary FW[1]	✗	Numerical
Ensemble learning	Soft	XSSC	-	✗	Numerical

[1] Feature Weighting
* N/A: not available

3.5 Multi-Objective Clustering

Clustering algorithms provide a partition of the data based on one cluster validity measure. However, assuming a homogeneous similarity measure over the entire data set makes algorithms not robust to variations in cluster shape, size, dimensionality, and other characteristics [Handl and Knowles, 2007]. Therefore, it is beneficial to optimize multiple validity indices simultaneously to capture different aspects of the datasets. The goal of Multi-Objective clustering methods (MOC) [Law et al., 2004] is to derive significant clusters by applying two or more objective functions. It aims to optimize tradeoff among multiple objectives under certain constraints simultaneously. MOC methods use a two-step process: 1) generate various clustering solutions and store the Pareto-optimal. , 2) construct an optimal partition based on the Pareto-set solutions. It is different than ensemble clustering, which operates with similar objective functions. Therefore, relevant clusters may be influenced by weak ones.

Multiobjective optimization only focuses on a small number of solutions which are not dominated by other solutions. These nondominated solutions are recorded in Pareto optimal solutions. The goal of the multi-objective problem is to find a set of solutions as close as possible to the solution in Pareto optimal solutions and as diverse as possible. We present the definitions of some definitions in the following.

Definitions:

- *Dominated solutions:* a solution X is said to dominate a solution Y if $\forall j = 1, 2, ..., m, f_j(X) \leq f_j(Y)$, and there exists $k \in 1, 2, ..., m$ such that $f_k(X) < f_k(Y)$.
- *Pareto-optimal solutions:* a solution X is called Pareto-optimal if it is not dominated by any other feasible solutions. The set of non-dominated solutions is called Pareto-set.

In the past decade, multi-objective evolutionary algorithms have been heavily used for the clustering problem. However, there has been no dedicated effort to review all of these methods. The most prominent effort in this direction can be found in [Mukhopadhyay et al., 2015], in which many multi-objective clustering algorithms and techniques were presented.
This chapter presents a thorough survey of the state-of-the-art for a wide range of multi-objective clustering algorithms and also some MOC concepts and techniques.

Fundamental Concepts

Most of the Multi-Objective clustering methods use an evolutionary representation for the clustering solutions as their use of population enables the variation of solutions and makes it easier to keep a population of clustering solutions and apply genetic operators. However, the use of such representation requires the following concepts:

- Choosing an evolutionary encoding to represent a clustering solution.
- The generation of the initial population by an effective initialization scheme.
- Suitable genetic operators to variate the solutions.
- Choosing two or more objective functions as a fitness function to choose the non-dominated solutions.
- Developing a technique to obtaining a single clustering solution for the Pareto-set (leader selection method).

The choice of these components is crucial for the clustering quality and the algorithm scalability. In the next sections, we present some examples used in the state of the art of the components presented above.

Evolutionary Encodings

Many representations were presented in the previous MOC methods [Mukhopadhyay et al., 2015]. We describe some of them in this section:

- **Centroid-based representation:** In this representation, the cluster center is represented by an array of d real numbers representing its coordinates, where d is the dimension of the dataset (number of features). After each iteration, only the centroid is updated using the points assigned to it.
- **Locus-based representation:** Proposed by [Handl and Knowles, 2007], it represents each clustering solution by a graph where the edges are the links between data points. The graph is represented by an array of n (number of points), each value is an integer, if two points have the same number, they are connected by an edge, the data points contained in the same connected component will then belong to the same cluster.
- **Point-based representation:** In this representation, the clustering solution is represented by an array of n (number of points) integers, each value represents the cluster label of the particular data point. Thus, if the value i of the array equals k, then the i^{th} data point belongs to cluster k.

Initialization Schemes

The initialization process is the first, and the most important step in the multi-objective methods as a good scheme can lead to faster convergence, while a bad scheme can lead to bad final solutions. In most of the multi-objective clustering methods, the clustering solutions in the initial population have been generated randomly, and this random initialization depends on the type of encoding used in the algorithm (Cluster centers, assignment..). Some other techniques have been used in some algorithms like the initialization using Voronoi diagrams [Handl and Knowles, 2004],

K-means clustering algorithm [Handl and Knowles, 2007], ACO and PSO [Handl and Meyer, 2007, Gong et al., 2017]. The primary objective of this type of initialization is to replace the random initial population by good clustering solutions generated by these methods.

Genetic Operators

Genetic operators are essential for MOC methods as they enable the variety and diversity of the clustering solutions. We list in the following the most used genetic operators:

- **Selection operator:** Selection operators are for generating a mating pool of clustering solutions. It gives preference to better solutions, allowing them to pass to the next generation of the algorithm. The best solutions are determined using an objective function. In MOC methods, all the objective functions participate in electing the best solutions called Pareto-optimal. The mating pool generated by the selection operator is then used by the next operators or by another algorithm.

- **Crossover operator:** Crossover operation is used for exchanging genetic information among the clustering solutions in the mating pool. The type of crossover used depends on the representation of the clustering solutions (Section 3.5), we list some of the well-known crossover techniques in the following:

  · **Single point crossover:** a crossover point i chose randomly, and the resulted chromosome is composed of values from the beginning to i of the first parent and from i+1 to the end of the second parent.

  · **Two point crossover:** two random crossover points are selected, values from the beginning of chromosome to the first crossover point are copied from one parent, the values from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent.

  · **Uniform crossover:** values are randomly selected from the first or the second parent

  · **Arithmetic crossover:** some arithmetic operation (AND, OR,..) is performed to make a new offspring.
  The different types of crossover are illustrated in Figure (3.12)

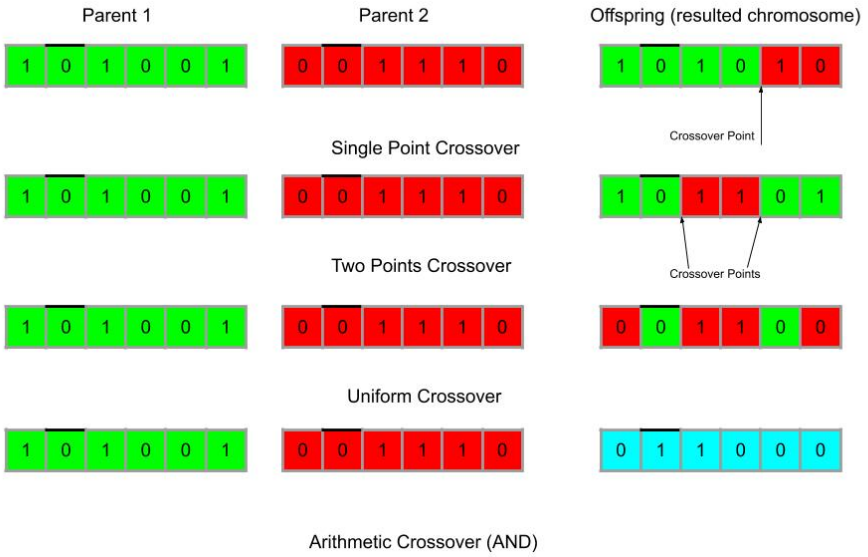


Figure 3.12 – *Different types of Crossover operator.*

- **Mutation operator:**

  · **Bit Flip Mutation:** A value is selected randomly from a chromosome and flipped (0 becomes 1, and 1 becomes 0). This is used for binary-encoded chromosomes.

  · **Random Resetting:** Random Resetting is an extension of the bit flip for the integer representation. Instead of swapping the value, it gives a random value from a pre-defined range to the randomly selected position.

  · **Swap Mutation:** Two positions are randomly selected, and their values are interchanged.

  · **Scramble Mutation:** A subset of the chromosome of random length is selected, and its values are scrambled or shuffled randomly.

  · **Inversion Mutation:** It is similar to Scramble Mutation, but instead of shuffling the selected subset, it completely inverses the values.

The different types of mutation are illustrated in Figure (3.13)

Figure 3.13 – *Different types of Mutation operator.*

Objective Functions

One of the important aspects of MOC is the choice of suitable objective functions that are to be optimized simultaneously. For each clustering solution, several quality measures exist, and the goal is to maximize inter-cluster similarity and minimize intra-cluster similarity. To satisfy these requirements, we introduce two objective functions *compactness* and *separateness*. More objective functions are described in Section 3.2.

- *Compactness*: the compactness of a clustering solution reflects the overall intra-cluster size of the data and has to be minimized. The compactness of a clustering solution is computed as follows:

$$Compactness_{\mathcal{C}} = \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} \delta(\mathbf{x}_i, \mathbf{w}_{\phi(\mathbf{x}_i)}) \tag{3.15}$$

  Where $\mathcal{X}$ is the dataset and $\phi(\mathbf{x}_i)$ is the index of the cluster where $\mathbf{x}_i$ belongs. $\delta(\mathbf{x}, \mathbf{w}_{\phi(\mathbf{x}_i)})$ is the euclidean distance between the data point $\mathbf{x}$ and $\mathbf{w}_{\phi(\mathbf{x}_i)}$.

- *Separateness*: the separateness of a clustering solution is the mean distance between clusters. It reflects the inter-cluster similarity and should be maximized. The separateness of a cluster is the shortest distance between a data point in this cluster and another data point of his neighborhood belonging to another cluster. The separateness is computed as follows:

$$Separateness_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} (min_{\mathbf{x}_i \in c, \mathbf{x}_j \in \mathcal{K}_i, \mathbf{x}_j \notin c} \delta(\mathbf{x}_i, \mathbf{x}_j)) \tag{3.16}$$

Where $\mathcal{K}_i$ is the neighborhood of the data point $\mathbf{x}_i$ belonging to the cluster c.

Solution Selection Methods

Optimizing multiple clustering objectives ought to produce a set of non-dominated solutions. The most suitable solution that indicates the final clustering result should be retrieved with the help of an expert. The methods in literature usually select the solution concerning (1) the performance of the solutions in terms of internal/external validity indices, or (2) the shape of the Pareto set, which depends on the value of the objectives.

Multi-Objective Clustering Algorithms

This section discusses previous works on multi-objective clustering problems and highlights the most relevant algorithms proposed in the literature to deal with these problems.

MOCK [Handl and Knowles, 2007] Multi-objective clustering with automatic K-determination, consists of two main phases: In its initial clustering phase, MOCK uses a Multi-Objective Evolutionary algorithm (MOEA) to optimize two complementary clustering objectives. The output of this first phase is a set of a mutually non-dominated clustering solution. Each corresponds to different tradeoffs between the two objectives. In the second phase, MOCK analyzes the shape of the tradeoff curve. It compares it to the tradeoffs obtained for an appropriate null model (i.e., by clustering random data). Based on this analysis, the algorithm provides an estimate of the quality of all individual clustering solutions and determines a set of potentially promising clustering solutions. Often, a single solution is preferred, and, in these cases, the number of clusters inherent to the data set, k, is thus estimated implicitly. Figure (3.14) illustrates the pareto set in MOCK algorithm. The improved version of MOCK, Δ-MOCK has been proposed [Garza-Fabre et al., 2017], which can significantly decrease the computational overhand and reduce the search space.

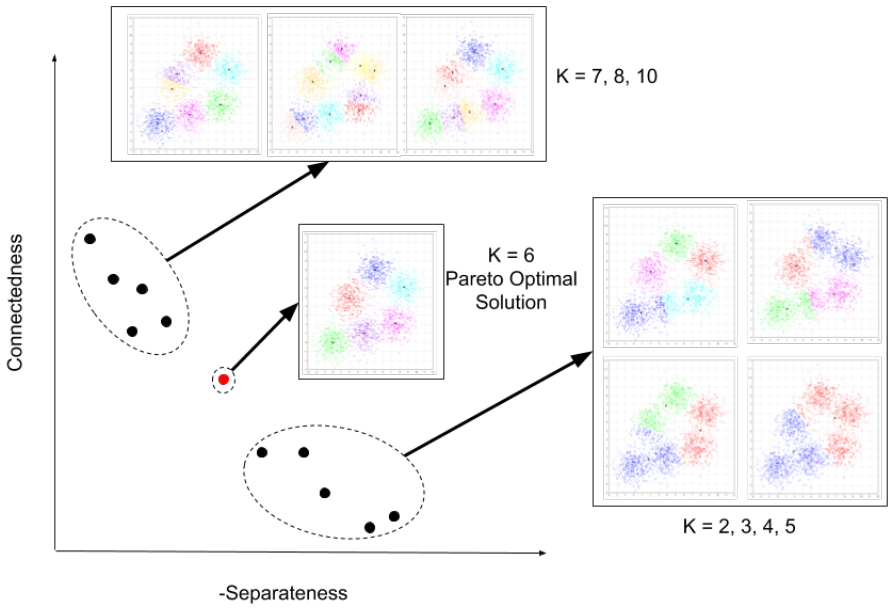


Figure 3.14 – *Clustering solutions plotted according to their objective functions. Each point represents a clustering solution. The pareto optimal solution is obtained when K=6.*

An Ant Colony Optimization-based clustering method ACO-C [İnkaya et al., 2015] combines the connectivity, proximity, density, and distance information with the exploration and exploitation capabilities of ACO in a multi-objective framework. The proposed clustering methodology is capable of handling several challenging issues of the clustering problem, including solution evaluation, extraction of local properties, scalability, and the clustering task itself.

Multi-objective evolutionary algorithms with simultaneous clustering and classification MOASCC [Luo et al., 2016] uses a clustering process to enhance the performance of the classification. To achieve this goal, two objective functions, fuzzy clustering connectedness function, and classification error rate, are adopted. A mutation operator is designed to make use of the feedback from both clustering and classification.

IMCPSO [Gong et al., 2017] proposes an improved multi-objective clustering framework using particle swarm optimization. The authors used the overall deviation and mean distance between clusters as objective functions. They introduced a clustering method to improve each particle (clustering solution) by finding a topological center, which is the point that has the maximum neighbors belonging to the same cluster Figure (3.15) illustrates the using of topological centers to improve the clustering. Finally, the best particle is selected from the Pareto-set based on the sparsity of the solution.

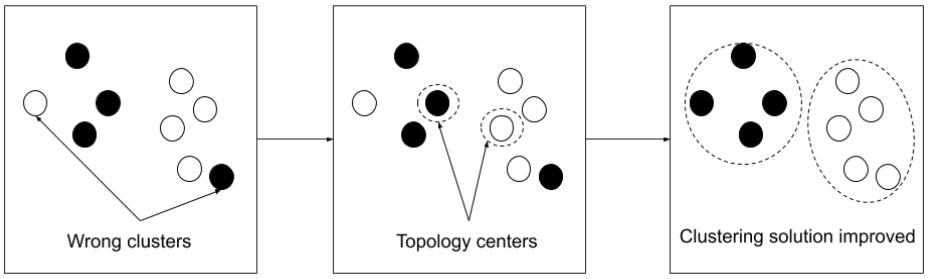


Figure 3.15 – *Using topology centers to improve the clustering solution.*

EMO-KC [Wang et al., 2018] uses the term bi-objective clustering to describe a MOC method with two objective functions. The method has two main steps (i) constructing two conflicting objective functions, and (ii) solving the bi-objective optimization problem with an effective EMO(Evolutionary Multi-Objective) algorithm.

Another MOC algorithm, SOMDEA-clust [Saini et al., 2019], proposes an efficient automated decomposition-based multi-objective clustering technique, which is a hybridization of Self-Organizing Maps (SOM) and differential evolution algorithm. Two internal cluster validity indices, namely, Silhouette index (SI) and PBM (Pakhira-Bandyopadhyay-Maulik) index, are used as objective functions. SOM algorithm is used to creates new solutions based on the neighborhood of each neuron. AMOGA [Dutta et al., 2019] Automatic clustering by a multi-objective genetic algorithm is a Multi-objective clustering algorithm that handles numeric and categorical features. Each clustering solution is encoded as a gene to apply Genetic operators. The method initializes a population using K-prototypes algorithm and GA operators crossover and mutation. AMOGA uses compactness and separateness as objective functions and different validity measures (DB index, Purity...) to select the best solution from the Pareto-optimal set.

Multi-objective Gradient Evolution algorithm [Kuo and Zulvia, 2020] extends the Gradient Evolution GE algorithm, so then it can be applied for the multi-objective problem. This paper applies the Pareto ranking assignment to sort the vectors based on their fitness values. K-means is then used to perform a final clustering on the Pareto-optimal solutions to obtain the final clustering.

Combinatorial Multi-Objective Pigeon Optimization algorithm (CMOPIO) [Chen et al., 2020] is based on a bio-inspired algorithm called Pigeon Optimization PIO. In CMOPIO, pigeons only interact with the pigeons in their neighborhood. Meanwhile, the update of the pigeon's position and velocity relies on each pigeon's neighborhood rather than the global best position. These improvements allow the CMOPIO to identify a variety of Pareto optimal clustering solutions.
Table (3.4) Compares the Multi-Objective Clustering Algorithms.

Table 3.4 – *Comparison Between Multi-Objective Clustering Algorithms*

Method	Encoding scheme	Genetic functions	Objective functions
MOGE	Centroid-based	/	SSW(Separateness) SSB(Compactness)
CMOPIO	Locus-based	/	Connectivity Compactness
MOCK	Locus-based	/	Stability
Δ-MOCK	Centroid-based	Crossover Mutation	Connectivity Class error rate
ACO-C	Point-based	/	Adjusted Compactness Relative Separateness
MCPSO	Locus-based	/	Compactness Separateness
SOMDEA-Clust	Centroid-based	Mutation Crossover	PBM Index Silhouette Index
IMCPSO	Locus-based	/	Overall deviation Mean distance between clusters
MOEASCC	Centroid-based	Mutation	J_{In} (Connectedness) J_{Add} (Error rate)
EMO-KC	Centroid-based	Crossover Mutation	SSD Overlap-Separateness

3.6 Conclusion

In this chapter, we have studied the concepts of clustering and the main existing classical methods. We presented the most used similarity and validity measures, which are crucial when constructing and validating a clustering algorithm.
As data clustering has attracted a significant amount of research attention, many clustering algorithms have been proposed in the past decades. We provided a survey of the most used clustering algorithms for each clustering category. We compared these clustering categories and presented some merits and limitations of each one of them. We also provided details of some subspace and multi-objective clustering algorithms and demonstrated their efficiency in clustering high dimensional data and discovering arbitrary shaped clusters.

Stream Data Analysis 4

In this chapter, we discuss the clustering of stream data and its main properties and techniques. We outline an overview of the clustering stream data methods and provide a comparison between these methods.

Introduction

Recent advances in both software and hardware technologies have allowed the acquisition of massive amounts of data continuously. This data grows faster than our ability to store or process it, but analyzing it can lead to interesting information that can be helpful in various fields such as financial transactions, telephone records, sensor network monitoring, telecommunications, website analysis, weather monitoring, and e-business. However, the analysis of large scale data leads to some big challenges: with the evolving data, it is not possible to process the data by using multiple passes efficiently. Instead, only one pass over the data is possible. This leads to constraints on the implementation of the stream algorithms. Other restrictions have to be considered by algorithms when processing data streams. We list some of them in the following:

1. Non-stationary and (potentially) infinite data points.
2. The order of data points needs to be respected when processing data streams.
3. The size of a stream is (potentially) unbounded.
4. Data points can not be stored and need to be processed in one pass.
5. The probability distribution may change over time.

Data streams are ordered and potentially infinite sequences of data points created by a typically non-stationary data generating process. The storage of this data is non-possible, also random access. only one (or few) passes possible through the data. Figure 4.1 illustrates the process of the data stream clustering.

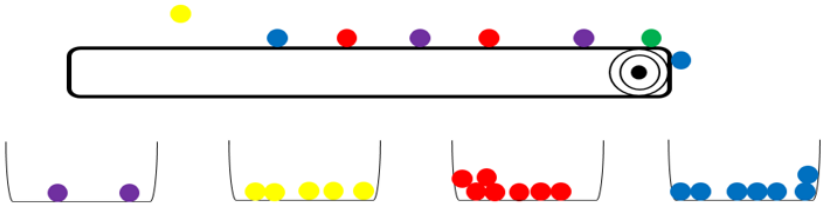


Figure 4.1 – *Stream clustering illustration*

Data stream analysis is the process of finding a complex structure within a large volume of data where the data evolves and arrives in an unbounded stream. Algorithms for Clustering Data Stream must consider restrictions in execution time and memory. Many algorithms use two phases. An online phase processes data stream points and produces summary statistics, and an offline component phase uses the summary data to generate the clusters. An alternative solution proposes to create final clusters without using the offline step.

To deal with clustering data stream restrictions, several classical methods have been modified, and many concepts have been introduced. In the following sections, we present some definitions and techniques that have been introduced in previous works.

4.1 Processing Step

As mentioned above, scientists face many constraints when dealing with data streams. The data stream can only be processed in one pass and must be analyzed following its order while respecting time and memory restrictions. In the following sections, we present some techniques to deal with data stream:

One pass processing

Since data streams can not be stored and are rapidly evolving, they can only be processed in one pass. The clustering solution is obtained by scanning data streams only once with the assumption that data objects arrive in chunks. STREAM [O'callaghan et al., 2002], which partitions the input stream into chunks and computes (for each chunk) a cluster. In most of the clustering algorithms, data points are processed in the order of their arrival. Data points are assigned to clusters, and clusters can be created, merged, or deleted over time. Figure 4.2 illustrates the different methods used in the one-pass clustering.

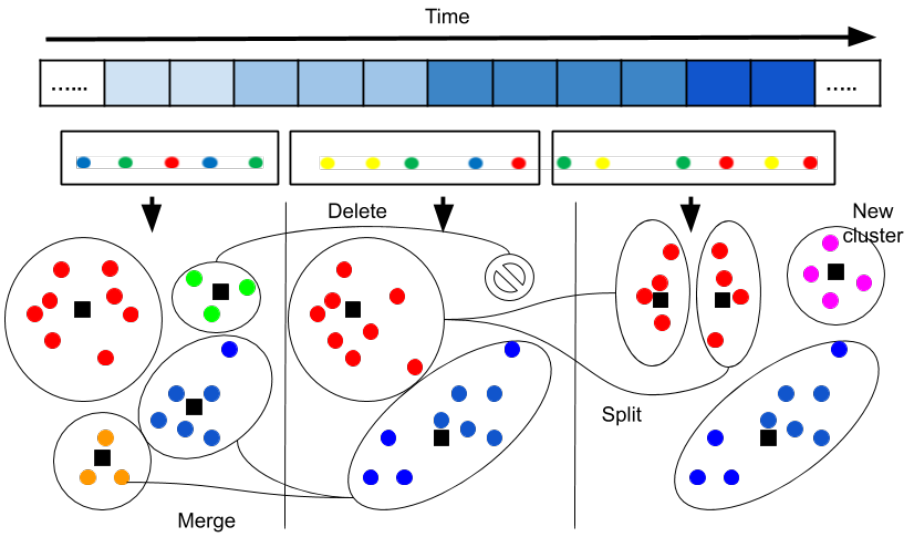


Figure 4.2 – *Cluster operations in one pass clustering*

Online and offline phases

Dynamic clustering over all possible time horizons of data streams cannot be done. Therefore, a large number of algorithms rely on two phases. An online phase process data stream points and produces summary statistics. Then an offline component phase uses the summary data to generate the clusters [Ntoutsi et al., 2012a, Ren and Ma, 2009a, Shukla et al., 2017a]. Alternative solutions also propose to create final clusters without using an offline phase [Lu et al., 2005, Khan et al., 2016b]. Figure (4.3) illustrates the online-offline process.

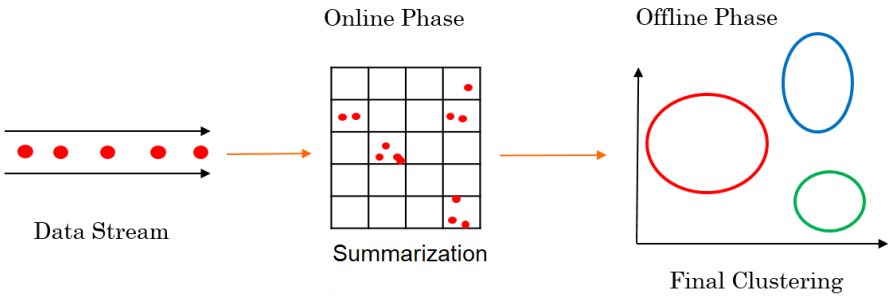


Figure 4.3 – *Online and Offline in Stream clustering*

Time Windows

The data streams rarely show stable distributions and they rapidly evolve. This method's basic idea is to divide the time-space into intervals called time windows and consider only the most recent data for clustering. There are four types of time windows:

- **Sliding Window**
  In this type of windows, only the recent data is considered. The observations are manipulated based on the principles of queue processing, where the first observation added to the queue will be the

first one to be removed. There are two types of sliding windows: 1- time-based sliding windows time is divided into equi-length intervals. 2- count-based sliding windows are defined by the number of incoming points.

- **Landmark Window**
  This sort of windows selects a fixed point in time called Landmark. Only the data points coming after this landmark are considered for the clustering. This model is not suitable for data stream since data grows very fast.

- **Damped Window**
  Unlike the other types of windows, the Damped windows associates weights with the data in the stream, and gives higher weights to recent data than those in the past. this helps to not discard the older data but, recent data will always have a better influence in computation.

- **Pyramidal Window**
  The pyramidal time window uses different granularity levels based on the recency of the data. This approach summarizes recent data more accurately, whereas older data is gradually aggregated.

Figure (4.4) illustrate the different models of time windows.

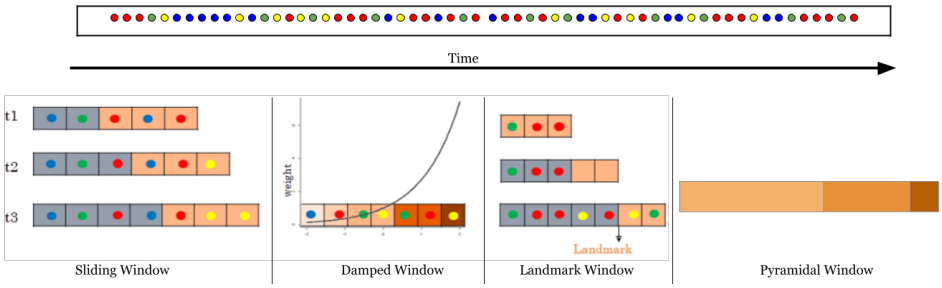


Figure 4.4 – *Time windows models*

Dimensionality reduction of data stream

Because of the inherently temporal nature of data streams, dimensionality reduction and forecasting are particularly important. When there are many simultaneous data streams, we can use the correlations between different data streams to make significant predictions on the future behavior of the data.

SPIRIT algorithm [Aggarwal, 2007], explores the relationship between dimensionality reduction and forecasting in data streams. The chapter also explores the use of a compact number of hidden variables to describe the data stream comprehensively. This close representation can also be used for effective forecasting of the data streams.

Load-shedding

Most of data stream comes from an external application; the rate of data cannot be controlled. Therefore, it is important for the application to have

the ability to quickly adjust to varying incoming stream processing rates [Aggarwal, 2007]. Load shedding gives the ability to degrade performance when the system lacks resources by getting rid of non-processed data points.

4.2 Summarization Step

Processing massive amounts of data streams require some space and time constraints on the computation process. Since data streams can be infinite, it is not possible to store the data. Therefore, the summaries of the data are constructed and held instead of the whole dataset. In the offline phase of the algorithm, the final clustering solution is computed in this summary. Many types of summaries have been presented in the literature. Each of them depends on a particular algorithm. We present some of these summaries in the following sections:

Random Sampling

Random Sampling is a method that allows us to constructs a synopsis of the data. Rather than deal with the entire data stream, this sample is constructed uniformly, and it represents the original data stream. Since random sampling methods require to know the length of the data in advance, they are not adapted for data streams. A modified approach called *Reservoir Sampling* [Vitter, 1985] is used for this kind of data. A set of data is maintained in a *reservoir* from which a random sample of size s can be generated. Each point has a probability of p_i; every new incoming point of the data stream has a probability of replacing an old random point in the reservoir. Min-Wise Sampling [Itoh et al., 2003] is based on assigning a random value in range 0 to 1 to a subset of samples m. When the system retrieves m elements, we select the sample with the minimum value. Figure 4.5 illustrates the process of Random Sampling.

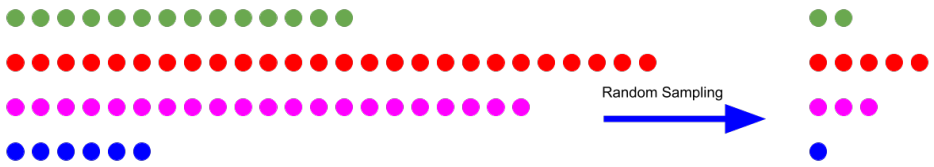


Figure 4.5 – *Random sampling example*

Sketching

Sketching means to generate uniform samples (sketches) of large masses of data. A sketch is a data structure that is used as a compressed representation of the database called a linear sketch. A linear sketch is a single vector containing various information about the database, which can be retrieved by the user. The sketch of a union of two Databases is the sum of their sketches. This property make linear sketches adapted to streaming data. There are two famous sketching methods: Count-Min Sketch and

Bloom Filter. We describe some sketching algorithms based on these two methods in the following sections.

- **Count-Min Sketch** Count-Min Sketch [Cormode and Muthukrishnan, 2005] summarize a data stream into a sketch, the method keeps an array of $d * w$ for each row a hash function h_j is associated to update the frequencies in each cell. when a new point $x_i(i_t, c_t)$ arrives, each hash function calculates its position $h_a(i_t) = j$, then each cell (a, j) is incremented.

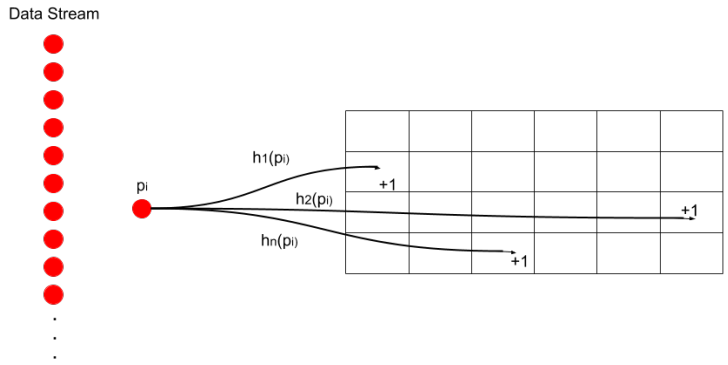


Figure 4.6 – *Count-Min Sketching example*

CSketch [Aggarwal, 2009] is a data stream partitioning clustering algorithm. It uses sketches obtained by the Count-Min method [Aggarwal, 2009]. The algorithm assigns each incoming point to the cluster with the most similar centroid. The similarity measure called dot-product, with $D^j(\overline{X_i})$ the dot-product of the incoming point with the cluster j. $q_r^j(x_i^r)$ is the frequency of the value x_i^r in the cluster j and m_j is the of points in the cluster j.

After choosing the centroid with the largest dot-product, all the frequencies in the chosen sketches are incremented for each dimension d. This algorithm is suitable only for qualitative data since the count-min calculates the frequency of each distinct value encountered.

UEStream [Chen et al., 2013] uses a modified count-min to summarize the uncertain data stream; it uses a metric [Anceaume and Busnel, 2013] to estimate the similarity between two streams of data using only the sketches. The similarity measure is called the Kullback-Leibler divergence and it is used to calculate the statistical difference between the data streams. The clustering is performed using a variance of K-means.

- Bloom Filter Bloom Filter is a very compact probabilistic data structure [Bloom, 1970] that allows us to know if an element is missing from a set and if it can be present in this set. The major disadvantage is that there can be false positives. The filter has two component: a boolean array T and k hash functions. To add an element, it just increments by 1 each case with index $j = h_i(e)$ for i between 1 and k.

XStreamCluster [Papapetrou and Chen, 2011] is used for clustering XML documents. These documents are represented as graphs with nodes that are represented by hash functions in the form of Bloom filters. Clusters are obtained by merging the bloom filters.

BloomStream [Sabau, 2016] is an algorithm based on grids, and each cluster is represented by a Bloom filter. The count-min sketch is used to represent a frequency table. Experiments have shown the effectiveness of this algorithm and its ability to detect outliers.

Micro Clusters

The Micro-Cluster is a structure that helps summarize large amounts of incoming data points without losing too much information. This structure can be hierarchical like the CF tree in BIRCH [Zhang et al., 1996] or a simple vector-like DenStream [Amini and Wah, 2010]. The micro-clusters can be incrementally updated as the data stream flows; the advantage of these structures is that it is adaptable to the evolution of data and also to the multi-dimensional data.

Histograms

Histograms are a good representative structure to summarize data. Histograms are used to approximate the frequency distribution of element values. They are widely used with static datasets, but their extensions to the stream framework is still a challenging task. Some techniques [Garofalakis et al., 2002] have proposed histograms for the incremental setting to handle evolving streams. However, they do not always work because the distribution of the instances is assumed to be uniform, which is not always true in reality [Bahri, 2020].

Wavelets

Wavelets are popular summarization techniques for image and signal processing. They are used for multi-resolution hierarchy structures over an input signal like stream data. Wavelets project each signal, which is a set of points, onto an orthogonal vector. Each signal reconstructed from the top few wavelet coefficients approximates the original signal better. Figure (4.7) present an example of wavelet decomposition from [Aggarwal, 2007].

Granularity	Averages	Decomposed Wavelets Coefficients
4	(8, 6, 2, 3, 6, 6, 5)	-
3	(7,2.5,5,5.5)	(1,-0.5,-1,0.5)
2	(4.75,5.25)	(2.25, -0.25)
1	(5)	(-0.25)

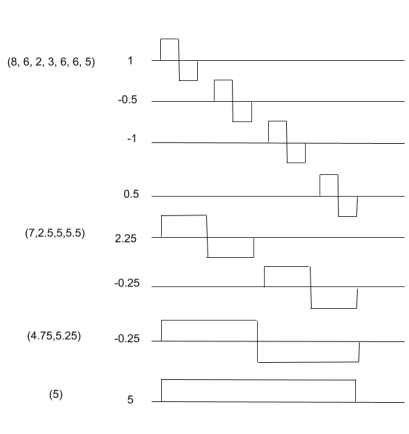


Figure 4.7 – *Wavelet decomposition example*

[Chen et al., 2010] uses W-HAS, which is a wavelet based hierarchical amnesic summary, as a summarization method. It consists of three steps:

1. Calculate the wavelet coefficient of each clustering center.
2. Extract data nodes from original data streams and calculating the wavelet coefficient of normalized data streams.
3. Dynamically update the wavelet coefficient of data streams and use the k-means method to get the final clusters.

4.3 Stream data clustering algorithms

In the next sections, we discuss some data stream clustering algorithms and present a comparison between these methods. The following clustering categories are defined in Section 3.

4.3.1 Partitionning algorithms

CluStream [Aggarwal, 2009] is a two-component clustering method that clusters data using two phases:

1. An online micro-clustering that summarizes the stream as micro-clusters. These micro-clusters are stored at snapshots in time that follow a pyramidal time frame.
2. An offline macro-clustering component that cluster these summaries into the final clusters.

Before presenting the algorithm's phases, we must define two concepts:
1) Micro-clusters: Statistical information about the data locality. The micro-cluster structure is a temporal extension of the cluster feature vector CF. A micro-cluster is a tuple (N, LS, SS, LST, SST) where:
- N number of data points.
- LS sum of the data points.
- SS squared sum of the data points.
- LST sum of the time stamps of the N data points.

- SST sum of squares of the time stamps of the N data points.

2) Pyramidal time frame: The micro-clusters are stored at time snapshots that are stored at different levels of granularity depending upon the recency. Snapshots are classified into different orders which can vary from 1 to log (T) where T = clock time elapsed since the beginning of the stream.

Figure (4.8) illustrates the difference between Micro and Macro clusters.

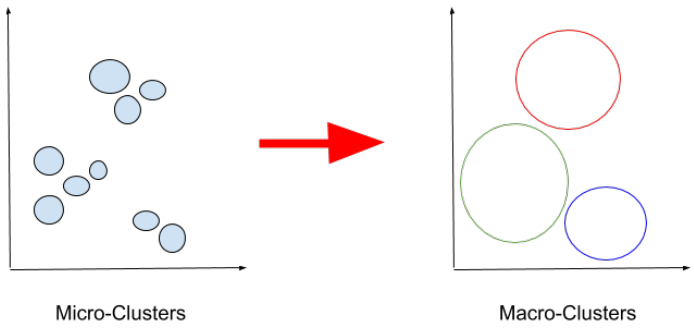


Figure 4.8 – *Difference between micro and macro clusters.*

Algorithm (3) presents the online phase of CluStream.

Algorithm 3 Online phase of the CluStream algorithm

input : q number of initial clusters, Initnumber: number of required initial points to start the clustering
output : Micro-clusters of points
- **Initialization:**
Wait until InitNumber points to arrive ;
Apply K-means algorithm to crate q clusters ;
while *there is a point p to process* **do**
  - Assign p to the closest cluster c ;
  if *p falls within the maximum boundary of c* **then**
    - p is absorbed by c ;
    - Update statistics of c ;
  else
    - Create a new micro-cluster with p, initialize its statistics ;
  - Keep q micro-clusters by Delete/Merge clusters ;

The micro-clusters snapshots are stored at particular times. These snapshots allow the user to search for clusters in different time horizons through a pyramidal time window concept. The offline step is applied on-demand upon the q maintained micro-clusters. A k macro-clusters is computed based on a time horizon h and using a clustering algorithm like K-means. CluStream is unable to find clusters with arbitrary shapes. It is incapable of detecting noise and outliers. It is also not suitable for large datastream.

RCD+ [Li et al., 2020] is a method of the partitioning data stream of

Relational Queries RQS using a clustering method. When users send an RQS, it is converted into topology tasks running on a stream processing system SPS. In an SPS, each node executes queries in parallel. We present in the following an example of a query to select information collected by a GPS system called GeoLife [Li et al., 2020]:

SELECT altitude, SUM (speed)/COUNT (*) FROM GEOLIFE GROUP BY latitude WINDOW (SLIDING, 10, 1)

To improve the query efficiency of SPSs and cope with data stream distribution skewness, the authors designed a granularity partitioning strategy that includes clustering partitioning (Clu-partitioning). Clu-partitioning is used to re-partition the data stream within and between nodes in three steps:

- According to the raw data volume and the parallelism of processing units, Clu-partitioning determines the number of units that need to move in and move out.
- It determines the number of clusters n.
- The STREAM algorithm is used to find the final clusters.

HP-Stream [Aggarwal et al., 2004] is a high-dimensional projected data stream clustering method. It is an extension of CluStream to handle high dimensional data. One projected clustering means that the algorithm selects only a subset of dimensions; the number of dimensions is not the same for each cluster. This is due to the fact that the relevance of each dimension in each cluster may differ. HP-Stream uses a Fading Cluster Structure (FCS) to stores the summary of streaming data, and it gives more importance to recent data by fading the old data with time. The main inconvenience of HP-Stream is his non-capacity to find clusters with arbitrary shape and the requirement of setting parameters like the number of clusters and the average number of projected dimensions parameters.

SWClustering [Zhou et al., 2008] is a clustering algorithm for evolving data streams over the sliding window; it introduces a new data structure called the Exponential Histogram of Cluster Features (EHCF). The exponential histogram is used to handle the in-cluster evolution, and the temporal cluster features represent the change of the cluster distribution. The proposed EHCF synopsis can provide sufficient information to calculate the final clusters. SWClustering is unable to find clusters with arbitrary shapes and its incapable of handling outliers.

Stream KM++ [Ackermann et al., 2012] is an extension of the K-Means clustering algorithm for handling stream data. It uses a random sampling method called the merge-and-reduce technique to obtain a small sketch from the data streams. It stores the data summaries in a data structure called a coreset tree, which is a binary tree in which each node is the union of all elements that descend from it. The advantage of such a corset is that we can apply any fast approximation algorithm (for the weighted problem) on the usually much smaller corset to compute an approximate solution for the original set more efficiently.

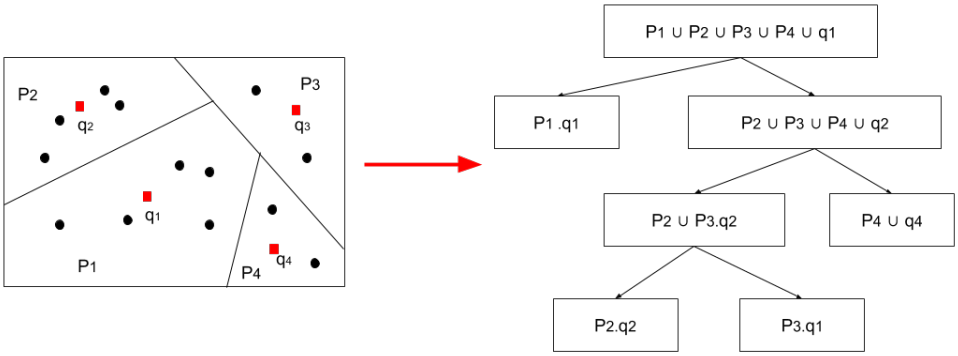


Figure 4.9 – *Coreset tree.*

In Algorithm (4) we illustrate the process of coreset construction.

Algorithm 4 Coreset construction

input : m: size of seed, P: a set of points
output : A coreset tree
Choose an initial coreset point v uniformly at random from P, w_v is the prototype of v;
Create two child v_1 and v_2 from the root as follow: Choose a data point x^j with probability $\frac{Dist(x^j,w_v)^2}{SSE_v}$ where SSE is the sum of squared distance;
Allocate data points x^i in v to the closest child between v_1 and v_2;
while *The number of nodes* $< m$ **do**
  Select a child with probability $\frac{SSE_{child}}{SSE_{parent}}$;
  Create two child v_1 and v_2 from the child as follow: Choose a data point x^j with probability $\frac{Dist(x^j,w_v)^2}{SSE_v}$;
  Allocate data points x^i in the selected child to the closest child between v_1 and v_2;

StreamKM++ maintain the data in L Buckets $B_1, B_2, .., B_L$, each Bucket have a size of m. When a data point arrive, it is assigned to the first available Bucket. If two adjacent Buckets are full, they are merged and a coreset tree is constructed reducing the $2m$ resulting points to m points. In the offline phase, k-means++ is used to find the final clusters.

StrAP [Zhang et al., 2008] is based on Affinity Propagation [Frey and Dueck, 2007] for data stream clustering. It introduces the notion of Reservoir to store potential outliers of data points. AP is an optimization algorithm based on message passing. For each arriving data point, StrAP either updates the model with it or puts it in the Reservoir. StrAP uses a Change Point Detection (CPD) test, called the Page Hinkley test, to catch drifting points that significantly deviate away. When the CPD test is triggered, the new model is rebuilt from the current model and data items in the Reservoir. The memory usage of STRAP is small and mainly depends on the number of outliers and exemplars, which may vary slightly in the streaming process. The clustering algorithms for data streams should be up to date so that clusters can be obtained at any time, as soon as the new data item arrives in the system. Algorithm (5) illustrates the process of StrAP algorithm.

Algorithm 5 StrAP algorithm

input : *DS*: a stream of points, ϵ: fit threshold
output : A set of clusters and their prototypes
- Apply AP on the first batch of data to produce the first clusters. Initialize Reservoir to empty;
while *There is a point* x_i *to proceed* **do**
  - Choose cluster c as the closest cluster to x_i ; **if** *dist* $(c,x_i)<\epsilon$ **then**
    - Update c and The model ;
  else
    - Put x_i in the Reservoir;
  if *CPD is triggered* **then**
    - Rebuild the model based on the current model and the data points in the Reservoir; - Set Reservoir to empty;

Adaptive Stream K-means [Puschmann et al., 2016] is a partitioning clustering algorithm with automatic K determination. It was proposed to cope with concept drift in the input data. The clustering process is composed of initialization and continuous clustering phases. In the initialization phase, Probability Density Functions (PDFs) are computed from the data using kernel density estimation KDE [Parzen, 1962]. The PDFs are split into equiprobable distributions to obtain small and dense cluster areas. The number of clusters K is then set as the number of areas in the PDF. The boundaries of these areas are called beta points. The middle points between two adjacent betas are computed and saved as initial centroids.

In the continuous clustering phase, an incoming point is clustered using the k-means algorithm. The standard deviation and expected value of the data with the current distribution are stored. These values are tracked during the process. Any change in the statistical properties of the data triggers a data drift detection. The initialization phase is then repeated to compute new centroids. The Adaptive Streaming k-means is presented in Algorithm 6.

Algorithm 6 Adaptive Streaming K-means

input : X: the data stream, l: length of data sequence used for initialization
output : C: A set of clusters
Initialization phase:
Calculate the PDFs using KDE for each point;
Calculate the turning points (statistical changes in the data);
Find beta points;
Calculate cluster centers C as the means between two adjacent betas;

Continuous clustering:
Run k-means with the set of centroids C;
if *change detected* **then**
  Run the initialization phase and compute new centroids;

Discussion

This section discusses the Merits and Limitation of partitioning stream clustering algorithms. We list the limitations and the merits of each algorithm in the following:

- **CluStream**: **-Merits:** - Clusters large evolving data streams - Can detect change - Provide flexibility in a real-time and changing environment. **-Limitations:** - Sensitive to noise - The number of micro-clusters need to be set.
- **HP-Stream**: **-Merits:** - Scalable algorithm - Gives more importance to recent data points by introducing a fading function**-Limitations:** - No concept drift detection - Outlier detection not very clear - Number of cluster parameter required.
- **SWClustering**: **-Merits:** - Can find arbitraily shaped clusters **-Limitations:** - Not scalable - Not suitable for high dimensional data - Can not detect outliers.
- **StreamKM++**: **-Merits:** - Easy to implement - The use of coreset tree make the algorithm faster by reducing the number of points from $2m$ to m. **-Limitations:** - Can not detect arbitrarily shaped clusters - The number of clusters is pre-defined.
- **StrAP**: **-Merits:** - Ability to detect outliers - Reduce the time complexity by processing only non-noisy data. **-Limitations:** - The parameter ϵ can affect the resulted clusters.

4.3.2 Hierarchical algorithms

SHC (Statistical Hierarchical Clustering) [Krleža et al., 2020] is single-phase clustering algorithm. it uses statistical inference on the input data stream to obtain statistical distributions that are constantly updated. SHC is capable of performing outlier detection, component population forming and updating, and clustering in the same step. This is enabled by the statistical agglomeration concept, which allows outliers and components to be agglomerated based on the statistical relations between them. Statistical agglomeration allows forming of new components from a set of outliers, assimilation of outliers by growing components, or merging two or more components under the same arbitrarily shaped cluster. The statistical agglomeration allows SHC to automatically define the number of clusters K.

SHC proposes a population evolution tracking on the component level. This is achieved through a novel concept of sub-clustering. To analyze and capture the statistical change in the component population, each component uses an additional child SHC instance to cluster the latest population members. The results of such component sub-clustering can be interpreted as population move or separation into several sub-populations, i.e., component drift or split.

On the cluster level, each component drift or split can result in a cluster split. Hence traceability is supported on the cluster level as well. All this is computationally more complex than current two-phase data stream clustering algorithms [Krleža et al., 2020].

A-Birch [Lorbeer et al., 2016] is based the BIRCH algorithm presented in Section 3.3.2. It uses an automatic threshold estimation using Gap Statistic. It does not require the pre-definition of the number of clusters. It does

not require the pre-definition of the number of clusters. To estimate K, A-BIRCH analyzes a small subset of the data and extracts parameters such as the cluster radius and the minimal cluster distance. These parameters are then used to calculate a threshold that results in the correct clustering of elements.

ODAC (Online Divisive-Agglomerative Clustering) [Rodrigues et al., 2006] is a top-down hierarchical clustering algorithm for the time-series data stream. It uses both agglomerative and divisive hierarchical methods. After building a tree-like hierarchy, the leaves of this tree are the final clusters. ODAC incrementally splits the clusters based on their diameter with the goal of reducing the intra-cluster similarity. The diameter of a cluster is the maximum distance between every two points in this cluster. After finding the two points that define the diameter and if the condition is met, the system assigns each point to a new cluster, and each point of the old cluster is assigned to the closest cluster between the new ones.

ODAC uses Pearson's correlation coefficient to calculate the similarity score between time series. The Pearson's correlation coefficient between time series a and b with n data points is defined as follows:

$$corr(a,b) = \frac{P - \frac{AB}{N}}{\sqrt{A_2 - \frac{A^2}{n}}\sqrt{B_2 - \frac{B^2}{n}}},$$

where $A = \sum a_i$, $B = \sum b_i$, $A_2 = \sum a_i^2$, $B_2 = \sum b_i^2$, $P = \sum a_i b_i$.

Given the correlation coefficient, the dissimilarity between a and b is defined as follows:

$$Diss(a,b) = \sqrt{\frac{1 - corr(a,b)}{2}}$$

ODAC applies the Hoeffding bounds test a leaf node for splitting. the Hoeffding bound helps in selecting the pair of data points in the cluster which represents the diameter of the cluster. When a splitting point is defined, the pivots are separated into two newly created clusters. ODAC assigns each of the remaining data point in the old cluster to the one of the two new ones.

ClusTree [Kranen et al., 2011] is a compact and self-adaptive index structure for maintaining the summary of the data coming via stream. It is a parameter-free algorithm capable of processing the stream in a single pass and with available memory. It also uses an anytime inserts concept to dynamically adapts to the speed of the data stream. This concept allows the algorithm to be able to give a result at any time.
ClusTree uses micro-clusters and CF trees similar to BIRCH explained in Section 3.3.2. As in [Kranen et al., 2011] the ClusTree have the following properties:

- An inner node nodes contains between m and M entries. Leaf nodes contain between l and L entries. The root has at least one entry.
- An entry in an inner node of a ClusTree stores:
  - a CF of the data point it summarizes.

  - a CF of the data points in the buffer, this one can be empty.
  - a pointer to its child node

- An entry in a leaf of a ClusTree stores a cluster feature of the object (s) it represents.
- A path from the root to any leaf node has always the same length (balanced).

Each new incoming data point is assigned to the closest subtree in the CluTree with respect to the euclidean distance. ClusTree uses a buffer as temporary storage for the data points when the insertion procedure is not possible. The temporary buffer entry is taken along as a hitchhiker to the leaf node at every new entry to the subtree. The hitchhiker is placed in the buffer of the corresponding split node, such that some other data point may carry it down.
In order to give better importance to the recent data points, ClusTree uses a decay factor λ. It attributes a weight for each data point following the decay function as follows:

$$\omega(\Delta t) = \beta^{-\lambda \Delta t}$$

where β is a pre-defined parameter.

The cluster features CFs of the tree are updated as follows:

$$n^{(t)} = \sum_{i=1}^{n} \omega(t - t_{s_i})$$
$$LS^{(t)} = \sum_{i=1}^{n} \sum_{i=1}^{n} \omega(t - t_{s_i}).x_i$$
$$SS^{(t)} = \sum_{i=1}^{n} \sum_{i=1}^{n} \omega(t - t_{s_i}).x_i^2$$

where t_{s_i} is the timestamp at which the data point x_i was added to the CF.

ClusTree introduces an aggregation For fast streams. Instead of inserting each data point at a time, it sum up m incoming data points and insert their aggregation to the tree. For slower stream settings, the idle times between data points are used to improve the quality of the resulting clustering are proposed.

E-Stream [Udommanetanakit et al., 2007] is an evolution-based technique for clustering stream data. Its associates a weight to each cluster. The weights decreases over time following a fading function as follow:

$$f(t) = 2^{-\lambda t}$$

when a cluster have a low weight, it becomes inactive. While an active cluster is a cluster that is receiving new incoming data.

E-Stream introduces a new structure for the clusters called *Fading Cluster Structure with Histogram (FCH)*. The algorithm uses the histogram of the cluster data points to find the splitting point of a cluster. The bin's range is calculated as the difference between the maximum and the minimum

values divided by α. The split operator is performed if a statistically significant valley is found between two histogram values peaks along any dimensions. The split is performed only on the active cluster.

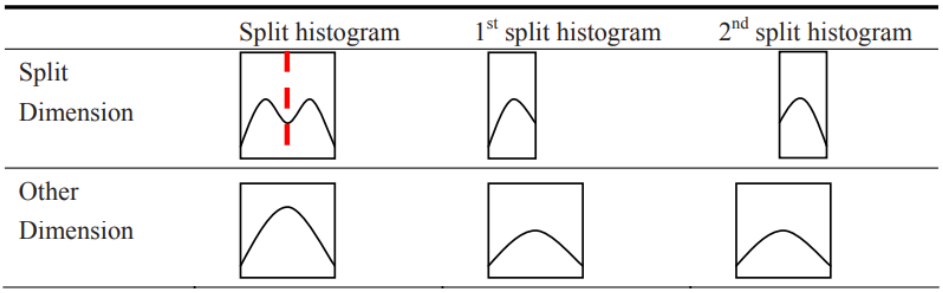


Figure 4.10 – *Histogram management in a split dimensionop and other dimension [Udommanetanakit et al., 2007]*

Every incoming point is either assigned to an existing cluster or a new cluster is created around it. If a cluster remains inactive for a certain time period, it may be deleted from the data space.

HUE-Stream [Meesuksabai et al., 2011] is an extension of E-Stream for heterogeneous data stream with uncertainty; it associates a histogram for both numerical and categorical attributes, for numerical data, HUE-Stream utilizes the same splitting criterion. For categorical attributes, the system chooses a splitting-attribute, i.e., that has significant frequency than the others within the same cluster, split-position is a position between a pair of adjacent values whose frequencies are the most different. A distance function with the probability distribution of two objects is presented to deal with uncertainty in categorical attributes. HUE-Stream detects the change in the data stream by using the distance function for merging clusters and finding the nearest cluster of the given new incoming data. The proposed histogram management is used for splitting clusters into categorical data.

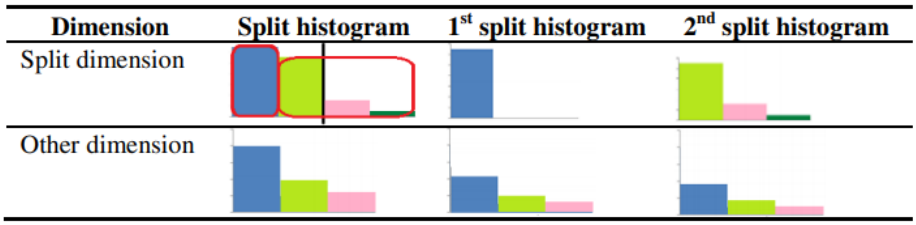


Figure 4.11 – *Histogram management in a split dimension and other dimension of categorical data [Meesuksabai et al., 2011]*

Discussion

This section discusses the Merits and Limitation of hierarchical stream clustering algorithms. We list the limitations and the merits of each algorithms in the following:

- **SHC**: - **Merits**:- Detects outliers - Detects arbitrary shaped clusters- Automatic K determination. - **Limitations**: - Memory and time consuming.

- **ODAC**: - **Merits**: - The time complexity don't depend on the on the number of features - Detects change and concept drift in the datastream - Automatic K determination.- **Limitations**: - Unable to detect arbitrary shaped clusters - Unable to detect noise
- **ClusTree**: - **Merits**: - Anytime clustering and self adaptive model size using buffer and hitchhiker concepts - Improves computational time by introducing an aggregation strategy and using the idle times with slow streams. - **Limitations**: - The use of K-means make the algorithm not able to detects clusters of arbitrary shapes
- **E-Stream**: - **Merits**: - Uses the fading function to adapt to the change - **Limitations**: - Requires many parameters to be specified by user.
- **HUE-Stream**: - **Merits**: - Adapted to both categorical and numerical data - Detects the change in the data stream. - **Limitations**: - Requires many parameters to be specified by user.

4.3.3 Density algorithms

DCDGA [Tareq and Sundararajan, 2020] is an online density-based method for clustering data stream using Genetic Algorithms GA. The GA is used to adjust suitable parameters for the cluster radius and minimum density threshold to cover the density clusters more accurately. DCDGA uses the Chebyshev distance to compute the radius of a micro-cluster and the distance between the data points. After the formations of the core micro-clusters, the final macro-clusters are constructed from the intersections of these CMCs.

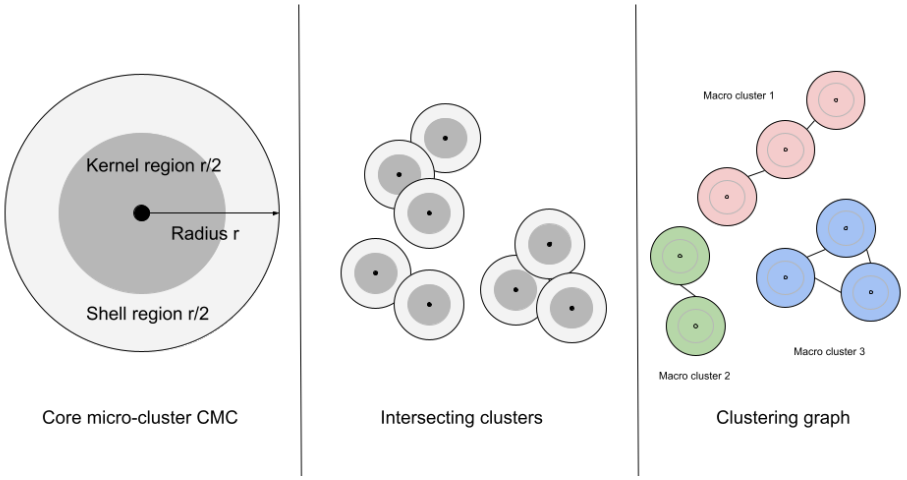


Figure 4.12 – *Formation of macro-clusters in the DCDGA algorithm*

For each data point, the DCDGA algorithm executes four steps:

- **Parameter optimization:** The DCDGA algorithm optimizes the radius and the threshold using a genetic algorithm. It initializes different chromosomes containing different pairs of the radius and the threshold between 0.01 and 1. Then, the crossover and mutation operators are applied. The fitness value is computed using the roulette-wheel method. The best pair of parameters is selected as the one

giving the highest fitness value. The fitness function is computed as follow:

$$fitval = 1 - (length(find(outputs == targets))/length(targets)),$$

where the outputs are the number of clusters generated, and the targets are the number of classes.

- **Assign core micro-clusters:** The new data point is assigned to the closest CMC using the Chebyshev distance. If the distance between the new data point and the closest CMC is inferior then a threshold, a new OMC is created with the new data point.
- **Remove weak CMCs:** if the CMC's life is under 0, it is removed with all its edges.
- **Update cluster graph:** If a CMC was changed, then the list of edges was changed too. The algorithm then creates a new macro-cluster throughout the graph.

CEC [Tareq et al., 2020] an online clustering algorithm based on density called the clustering of evolving data streams based on the adaptive Chebyshev distance. It keeps a graph of core micro-clusters CMC. Two linked CMCs are associated with the same macro-cluster. The CEC algorithm outlines the "minimum density threshold" for distinguishing the outliers from the CMCs along with the "decay" parameter for determining the data's evolving property. A CMC possessing a density lower than the minimum level is an outlier. Every time a new data point falls into CMC or helps create a fresh CMC.

The CEC algorithm has four steps:

- **Parameter selection**: the decay factor, the radius of a cluster and the threshold which is the minimum points that the outlier micro-cluster OMC need to be converted to a CMC.
- **Assign CMC**: If the distance between a point p and the nearest CMC or OMC is less then the threshold R, then p is assigned to it. Otherwise, a new OMC is created.
- **Kill cluster**: If the weight of a cluster is less then $\frac{1}{Decay}$, then the CMC is deleted.
- **Update cluster graph** If the list of CMC has changed then the graph need to be updated accordingly.

DenStream [Cao et al., 2006] is a density-based clustering algorithm that derives some of the concepts from the partition-based clustream and improves upon it. DenStream uses the damped window model with an exponential aging function. In the initialization phase, the DBSCAN algorithm is used on the first batch of data points. Two lists of micro-clusters are maintained during the online phase based on the DBSCAN paradigm: The potential micro-clusters PMC and the outliers micro-clusters OMC. The offline phase runs on-demand, the micro-cluster summaries are used

to produce the final clustering. In particular, each micro-cluster is considered as a data point, and a variant of DBSCAN is applied upon these data points.

Algorithm 7 DenStream algorithm

input : X: the data stream, ϵ, β, λ, μ
output : C: A set of macro-clusters
For each data points p ***in the stream***:
  Merge p into the closest PMC or OMC;
  if t *mod* T_p=o **then**
    for each PMC **do**
      if $w_p < \beta\mu$ **then**
        Delete c_p ;
    end for
    for each OMC **do**
      $\xi = \frac{2^{-\lambda(t-t_0+T_p)}-1}{2^{-\lambda T_p}-1}$;
      if $w_o < \epsilon$ **then**
        Delete c_o ;
    end for
if *a clustering request occurs* **then**
  Generating clusters;

SDStream [Ren and Ma, 2009b] is a density-based data stream clustering over sliding window, it has online and offline phases, it introduces the concepts of Temporal Cluster Feature (TCF) and Exponential Histogram of Cluster Feature (EHCF). The TCF contains the time scale information and the information of the feature, EHCF is a set of TCF. in the online phase EHCFs are used to store the p-micro-cluster and o-micro-cluster. Since the number of micro-clusters is limited, either a micro-cluster has to be deleted, or two clusters are merged. Outdated points are deleted via the temporal value t in the TCF if t doesn't belong to the bounds of the sliding window, TCF is deleted. In the offline step, a modified DBSCAN is used to get the final clusters of arbitrary shape, DBSCAN is also used to initialize a group of p-microclusters. The main usage of the exponential histogram is not clarified by the authors.

Stream OPTICS [Shukla et al., 2017b] is a variation of the density-based clustering algorithm OPTICS [Ankerst et al., 1999] which is based on the most basic density algorithm DBScan [Ester et al., 1996], the method can find clusters of arbitrary shape and can handle noise. The algorithm uses temporal windows to consider only recent data, different parameters like window size, threshold value, and radius are set by the user. The algorithm has an online and an offline, in the online phase DBScan is used to initialize the micro-clusters.

Discussion

This section discusses the Merits and Limitation of density-based stream clustering algorithms. We list the limitations and the merits of each algorithm in the following:

- **DCDGA**: **-Merits:** - Detects Noise - Optimal initialization of the parameters. **-Limitations:** - If the algorithm cannot merge the data point, it will create a new micro-cluster for each incoming data point - Genetic algorithm can make the algorithm costy in time and memory
- **CEC**: **-Merits:** - Detects Noise - Detects arbitrary shaped clusters **-Limitations:** - Higher time and memory complexity
- **DenStream**: **-Merits:** - Detects arbitrary shaped clusters - Detects concept drifts in data stream - Detects Noise **-Limitations:** - The number of micro-cluster can increase and exceed the memory limitation.
- **SDStream**: **-Merits:** - No assumption of the number of clusters **-Limitations:** - Can not handle High dimensional datasets
- **Stream OPTICS**: **-Merits:** - Can handle noise **-Limitations:** - Costy in time and memory - Can not handle High dimensional datasets.

4.3.4 Grid algorithms

CEDGM [Tareq et al., 2020] is an online density and grid based clustering algorithm for stream data. The first phase generates the Core Micro-Clusters (CMCs), and the second phase combines the CMCs into macro clusters. The grid-based method is used as an outlier buffer in order to handle multi-density data and noises. CEDGM forms grids by splitting the data space into small segments. Illustrative neighbor research is then performed on the grids to group them into cluster grids.

In the CEDGM, each CMC among radii $r_0/2$ contains a shell region r_0 and a kernel region $r \leq r_0/2$. Macro-clusters are formed by intersecting the shell region of CMCs and the kernel regions of other CMCs. The CMCs with a density that exceeds the minimum threshold but with no intersections are also considered macro-clusters. From the data stream, a new data point will fall into three regions. First, if the data point falls in a grid granularity space, it will create a new outlier. Second, if the data point falls in the CMC shell region can be assigned to the cluster and recursively update the CMC center and cluster count. Third, the data point allocated to the CMC, and the cluster count is updated when the data point falls in a kernel region. The created or modified CMC is examined to determine if the cluster density is greater than the minimum threshold. This CMC is then examined for new intersections with other CMCs. When new intersections are created, these CMCs are linked and assigned to the same macro-cluster. All connected CMCs must have the same macro-cluster and create an arbitrarily shaped cluster in an online manner.

DGStream [Ahmed et al., 2020] is an online-offline grid and density-based stream clustering algorithm. The online phase uses feature vectors represented by a micro-cluster for each grid to dynamically maintain the necessary information about the uninterrupted arriving data records. While in the offline phase, DGStream employs a DBSCAN algorithm to benefit from its speed and improve the running time. DGStream also uses a decay function mechanism to reflect the stream evolution process accurately.

DGStream employs a mechanism to detect the grids containing very few data points or do not receive new data points for long periods. The detected grid is deleted to maintain processing only with a limited number of dense grids, saving both time and memory of the system. DGStream also employs a mechanism to get rid of the noise and to handle outliers.

ExCC [Bhatnagar et al., 2014] is a clustering algorithm for the heterogeneous data stream. ExCC is a complete clustering algorithm which means, complete clustering is an approach in which an object is either a member of a cluster or an outlier. The exclusive clustering considers that one data point is a member of only one cluster. Figure 4.13 shows the Difference between exclusive and non-exclusive clustering.

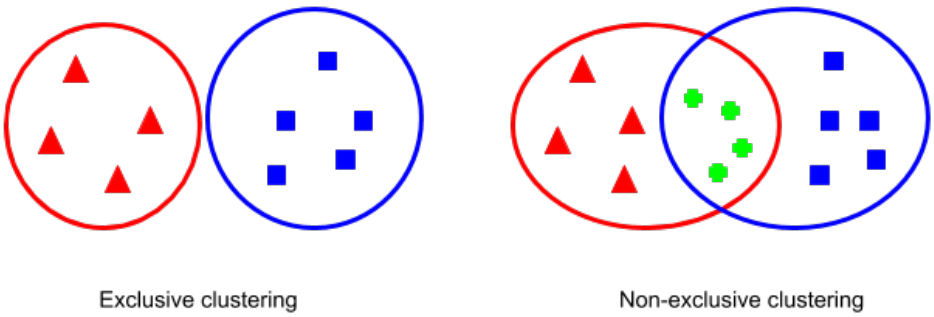


Figure 4.13 – *Difference between exclusive and non-exclusive clustering*

ExCC uses a grid structure for mixed attributes and assigns granularities for each attribute, according to the distinct values for categorical ones or to the number of equal-width intervals for numerical attributes. A speed-based grid pruning mechanism is employed by the ExCC algorithm rather than a window model such as fading one. Clusters that have not seen significant addition of data points since the last clustering are considered as old and have been removed. The ExCC algorithm has both online and offline phases and can detect noise. However, keeping grids requires more memory and time process.

DCU-Stream [Yang et al., 2012] algorithm is a density grid-based clustering algorithm over the uncertain data stream; it defines the concepts of the Adjacent grid and Core dense-grid. The adjacent grid of the current inspecting grid is the grid that has the common side, and the core dense-grid is the grid that has the best density and is surrounded by sparse-grids. The algorithm examines all grids to find core dense-grid, the neighbor grids, which are sparse, are considered as noise. DCUStream finds clusters with arbitrary shape, it outperforms many clustering methods, but it is time-consuming.

PKS-Stream [Ren et al., 2011] is a Density and grid based clustering

algorithm. the d-dimensional space is partitioned into small grids, then, to every incoming data point is assigned a a density coefficient calculated at each time t as follows:

$$d^{(t)}(x) = 2^{-\lambda(t-t_0)},$$

where λ is the decay factor and t_0 is the time of the beginning of the stream. PKS-stream introduces the Pkstree, where each node of the tree corresponds to a grid. The root of the Pks-tree contains a synopsis of the space S of the data. All the other nodes contain a grid synopsis of the data in a granularity i. The new incoming data point is mapped into the related cell in every level of the Pks-tree.

Pks-tree is used for recording the non-empty cells and also the relation between grids. If there is a grid cell for the data record, the data record is inserted. Otherwise, a new grid cell is created in the tree. In the offline phase, the algorithm clusters all the minimum cells located in the leaf-node level by assigning neighboring dense grids with the same cluster label.

DGClust [Gama et al., 2011] is a distributed clustering algorithm for data streams generated in sensor networks, which reduces both the dimensionality and the communication burdens. It allows every local sensor to retain an online discretization of the streaming data. The data stream is being incrementally discretized and sketched into a grid. Partitional incremental discretization (PID) is applied to every sensor to simplifies and summarizes the data stream then construct the final grid. DGClust performs with a fixed update time and space.

Discussion

This section discusses the Merits and Limitation of grid-based stream clustering algorithms. We list the limitations and the merits of each algorithm in the following:

- **CEDGM**: **-Merits:** - Finds arbitrary shaped clusters and detects outliers. **-Limitations:** - Several parameters to set - Depends on the parameters setting
- **PKS-Stream**: **-Merits:** - Can handle high-dimensional datasets - Improves computational complexity of the density algorithms - Depends on the parameters setting
- **DCUStream**: **-Merits:** - Lower time and memory complexity - Handles uncertain data. **-Limitations:** - Can not handle high dimensional datasets - Does not detect noise
- **ExCC**: **-Merits:** - Handles evolving data - Handles Noisy data - Clusters categorical and numeric data. **-Limitations:** - Can not handle high dimensional datasets - Limited time and memory.
- **DGClust**: **-Merits:** - The distribution ability makes the algorithm scalable **-Limitations:** - Unable to find arbitrarily shaped clusters.

4.3.5 Subspace algorithms

EDSSC [Sui et al., 2020] is a subspace clustering for high dimensional data streams. It can cope with the time-varying nature of subspaces underlying the evolving data streams, such as subspace emergence, disappearance, and recurrence. EDSSC is a two-phase algorithm: the algorithm stores a static summary called EDSSC summary in the first step. In the second phase, the average sparsity concentration index (ASCI) is proposed to promote clustering accuracy. EDSSC introduces a new method to automatically estimate the number of subspaces using singular-based Laplacian matrix decomposition. Assuming a similarity matrix $\mathbf{W}$ of the data matrix $\mathbf{X}_{dxN}$. The normalized Laplacian matrix is obtained as follows:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

where $\mathbf{D} = diag(\sum_{j=1}^{N}[\mathbf{W}]_{1j}, .., \sum_{j=1}^{N}[\mathbf{W}]_{Nj})$, $\sigma_{i=1}^{N}$ are the eigenvalues of $\mathbf{L}$. The estimation of the number of subspaces is calculated as follows:

$$k = max_i(|\phi_i|)_{i=2}^{N-1} - k_0$$

where $\phi_i = log_a(\sigma_i^2/\sigma_{i+1}\sigma_{i-1})$ and $a > 1$ is a constant, $k_0 = 0$ if $\sigma_{i+1}\sigma_{i-1} \geq \sigma_i^2$ and $k_0 = 1$ otherwise.

The initial clustering is obtained by performing the k-means algorithm on the first eigenvectors matrix. The first summary is initialized by the points selected following a random sampling approach. For each new incoming point, a decision is made whether this point is an outlier or a normal point. The algorithm follows the subspace evolution to detect its recurrence, emergence, or disappearance.

CashStream [Borutta et al., 2020] is a subspace algorithm for clustering data streams. CashStream uses the Hough transformation [Scheid and Schwarz, 2009] to perform an oriented subspace clustering. The Hough transformation originally has been introduced for detecting linear segments in image data by mapping every object in data space. It can identify the intersections of a specific amount of object functions. The related data objects are located on a line segment in data space if such an intersection exists.

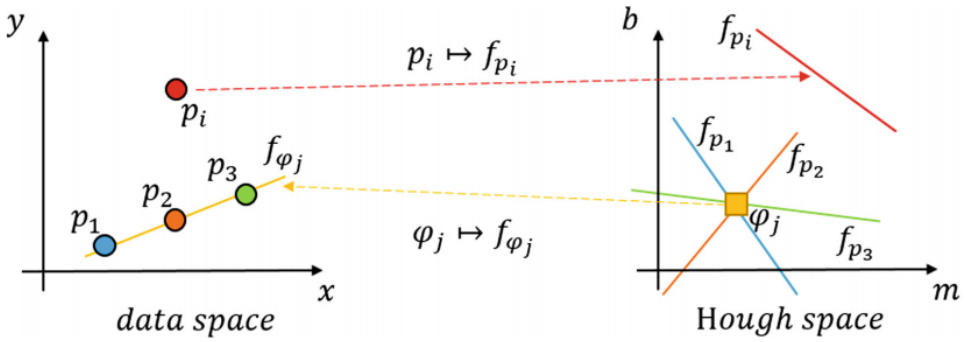


Figure 4.14 – *Left: data space, right: Hough space [Borutta et al., 2020]*

After transforming the data points from the data space to the Hough space, the algorithm finds the dense areas in the Hough space. It divides

the space into a grid and then for each cell c, if the number of object functions intersecting c is greater or equal than a pre-defined $minPts$ parameter, c is split. A grid cell c that is dense after $maxSplit$ divisions represents a cluster: the points corresponding to the functions intersecting c form a cluster within a arbitrarily oriented $(d-1)$-dimensional subspace. CashStream detects subspace clusters of lower dimensions by processing the resulting $(d-1)$-dimensional dataset recursively until no more cluster can be found.

CashStream introduces a new data structure to store the clusters called *Concept*. A concept consists of the following components [Borutta et al., 2020]:

- a set E containing $d-l$ equations in Hessian normal form,
- mean μ of all data objects that are assigned to the cluster,
- number of data objects N that are assigned to the cluster,
- the timestamp t of the last update
- reference P to parent Concept of dimensionality $l+1$, if $l \leq d$ - 1.

The algorithm uses an **Aging function** to give more importance to the recent data. The importance of a concept C is calculated as follows:

$$\mathcal{I}(C) = e^{-\lambda \Delta t}.N_C$$

where λ is the decay factor, Δt is the difference between current time and the timestamp of C. N_c is the number of data points assigned to C.
Two similar Concepts can also be merged using the **Unification function**. The resulted concept C^*, which is the result of the Unification of C_1 and C_2, is defined as follows:

- $E_i^* = \frac{\mathcal{I}(C_1).n_{E_{1,i}} + \mathcal{I}(C_2).n_{E_{2,i}}}{2}.x + \frac{\mathcal{I}(C_1).r_{E_{1,i}} + \mathcal{I}(C_2).r_{E_{2,i}}}{2}$
- $\mu_{C^*} = \frac{\mathcal{I}(C_1).\mu_{C_1} + \mathcal{I}(C_2).\mu_{C_2}}{2}$
- $N_{C^*} = N_{C_1} + N_{C_2}$
- $t_{C^*} = t_{C_1}$
- $parent(C^*) = parent(C_1)$

SubClusTree [Hassani et al., 2014] is an anytime grid-based subspace clustering version of LiarTree that also finds hidden clusters in the subspaces of the stream at any time. It can adapt to the different stream speeds and makes the best use of available time to provide a high-quality subspace clustering. Each subspace is represented by a Liar-tree, which is a structure where the micro-clusters are stored hierarchically. The micro-clusters in any higher level of the tree are bigger and less in number than the ones in lower levels. The most fine-grained microclusters are stored in the leaf level of the tree.

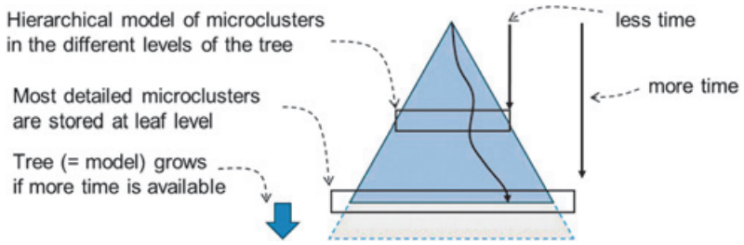


Figure 4.15 – *The main insertion and model concept of LiarTree [Hassani and Seidl, 2016]*

SubClusTree uses a forest of multiple liar-trees as its data structure. Each incoming data point is inserted into all one-dimensional trees. If the stream idles, the data point is also inserted into the next higher dimensional tree. Popular subspaces are decided over a certain batch using a heuristic that decides potential higher-dimensionality subspaces as the combination of popular lower-dimensional subspaces in an Apriori-like method. The heuristic used by SubClusTree estimates the density of flexible grids to efficiently distinguish the populated higher-dimensional subspaces from irrelevant ones [Hassani and Seidl, 2016].

HDDStream [Ntoutsi et al., 2012b] is the first algorithm for density-based projected clustering over high dimensional data streams; it introduces the notion of preferred dimension. A micro-cluster prefers a dimension if data points are denser along this dimension. This preference is controlled by a parameter Δ called variance threshold. A dimension preference vector of a micro-cluster is defined by the number of preferred dimensions. The micro-clusters with dimension preference vector is called projected micro-cluster, which shows that the micro-cluster is associated with only a subspace of feature space. The algorithm has online and offline phases. It uses an exponential fading function to remove expired points. The PreDeCon algorithm [Bohm et al., 2004] is used to extract the initial set of micro-clusters and to generate the final clusters in the offline phase. One of the inconveniences of this algorithm is that in the fading function, only micro-cluster weights are updated. The preferred vector should also be checked because it may change over time.

PreDeConStream [Hassani et al., 2012] improves the HDDStream algorithm by working on the offline phase. It uses a micro-clustering process Similar to **DenStream** [Amini and Wah, 2010]. This algorithm also introduces a subspace preference vector, which is defined based on the variance of micro-clusters and their neighbors. The subspace prefers vectors of the neighbors of newly inserted potential micro-clusters, as well as deleted potential micro-clusters, are updated and put in a list as affected micro-clusters. The affected micro-cluster list is used in the offline phase as expanding clusters to improve the efficiency of the offline phase.

Sibling Tree [Park and Lee, 2007] is a grid-based subspace clustering algorithm. It assigns for each feature a list of cells called a sibling list. Then the second level of lists monitors dense grids in two-dimensional subspaces. Successively, additional lists are used to monitor higher-dimensional cells. The algorithm incrementally updates the clus-

tering solution and uses a fading window model to give more importance to recent data.

Discussion

This section discusses the Merits and Limitation of subspace stream clustering algorithms. We list the limitations and the merits of each algorithm in the following:

- **EDSSC**: **-Merits:** - Adapts and detects the evolution of a subspace - Automatically detects the number of subspaces - Detects outliers **-Limitations:** - Depends on k-means in the initialization phase.
- **CashStream**: **-Merits:** - Oriented subspace detection - Finds clusters in arbitrary shapes **-Limitations:** - Costly in time and memory.
- **SubClusTree**: **-Merits:** - Handles High-dimensional data - Detects clusters in different-density data - Flexible to variying time allowances between data points **-Limitations:** - The use of liar-trees makes the algorithm not scalable.
- **HDDStream**: **-Merits:** - Detects ouliers **-Limitations:** - Unable to detect overlapping clusters and subspaces.
- **PreDeConStream**: **-Merits:** - Arbitrary shaped clusters detected - Adapts to the data stream evolution **-Limitations:** - Depends on the paramaters setting.

4.3.6 Evolutionary algorithms

evoStream [Carnein and Trautmann, 2018] (Evolutionary Stream Clustering) makes use of an evolutionary algorithm to bridge the gap between the online and offline components. Evolutionary algorithms are inspired by natural evolution where promising solutions are combined and slightly modified to create offsprings, which can yield an improved solution. By iteratively selecting the best solutions, an evolutionary pressure is created, which improves the result over time. evoStream uses this concept to enhance the macro-clusters through recombinations and small variations iteratively. Since macro-clusters are created incrementally, the evolutionary steps can be performed while the online components wait for new observations, i.e., when the algorithm would usually idle. As a result, the computational overhead of the offline part is removed, and clusters are available at any time. The online component is similar to DBSTREAM [Hahsler and Bolaños, 2016] but does not maintain a shared-density since it is not necessary for reclustering.

evoStream is based on **DBSTREAM** [Hahsler and Bolaños, 2016] (Density-based Stream Clustering), which uses the shared density between two micro-clusters to decide whether micro-clusters belong to the same macro-cluster. A new observation is merged into micro-clusters if it falls within the radius from their center. Subsequently, the centers of all clusters that absorb the observation are updated by moving the center towards x. If the point is not assigned to a cluster, it is used to initialize a new micro-cluster. Additionally, the algorithm maintains the shared density between

two micro-clusters as the density of points in the intersection of their radii, relative to the size of the intersection area. In regular intervals, it removes micro-clusters and shared densities whose weight decayed below a respective threshold. In the offline component, micro-clusters with high shared density are merged into the same cluster.

evoStream was used in [Supardi et al., 2020] to detect outliers in a data stream. The goal of this method is to treat the distinct data object as an outlier detection problem compared than the categorization problem.

HDCStream [Amini et al., 2014] (hybrid density-based clustering for data stream) first combined grid-based algorithms with the concept of distance-based algorithms. In particular, it maintains a grid where dense cells can become micro-clusters as known from distance-based algorithms (see Section 4). Each observation in the stream is assigned to its closest microcluster if it lies within a radius threshold. Otherwise, it is inserted into the grid instead. Once a grid-cell has accumulated sufficient density, its points are used to initialize a new micro-cluster. Finally, the cell is no longer maintained, as its information has been transferred to the micro-cluster. In regular intervals, all micro-clusters and cells are evaluated and removed if their density decayed below a respective threshold. Whenever a clustering request arrives, the microclusters are considered virtual points to apply DBSCAN [Ester et al., 1996]. The algorithm consists of three steps: (1) Merging or mapping: the new data point is added to an existing mini-cluster or mapped to the grid. (2) Pruning Grids and Mini-clusters: the grids cells, as well as mini-cluster weights, are periodically checked in pruning time. The periods are defined based on the minimum time for a mini-cluster to be converted to an outlier. The mini-clusters with weights less than a threshold are discarded. (3) Forming final clusters: final clusters are created based on mini-clusters, which are pruned. Each mini-cluster is clustered as a virtual point using a modified DBSCAN.

FlockStream is a bio-inspired algorithm for clustering data stream Kennedy [2006] simulating the behavior of a group of birds in flight. Boid is the abbreviation of the word bird-oid (which means in the form of a bird). These boids are interacting and follow certain rules:

- **cohesion** to form a group, the boids are getting closer to each other
- **separation** 2 boids can not be in the same place at the same time
- **alignment** to stay grouped, boids try to follow the same path

FlockStream uses agents to mimic the behavior of boids. Each point is associated with an agent. An agent can be of three types: basic, p-representative (potential micro cluster), or co-representative (outlier microcluster, it can become p-representative if adding points, its weight exceeds a threshold). In the initialization phase, a set of basic agents is deployed in In space, the agents that have a great similarity approach (cohesion) form a cluster, while the other agents separate. The Euclidean distance is used to calculate the dissimilarity between agents. Agents can leave one group to join another with more similar agents. at the end of this phase, a summary for each cluster is calculated, and the other two types

of agents appear p-representative and o-representative. In the second step, a mass of data stream is inserted. In this phase, the agents are updated as follows:

- if an o-representative or p-representative meets another representative, if their distance is less than a threshold, then they join to form a swarm (cluster)
- a basic agent A meets a representative R, if their calculated distance is lower than a threshold, A is absorbed by R
- a basic agent meets another, so if their similarity is less than a threshold, he joins to form an o-representative.

Discussion

This section discusses the Merits and Limitation of evolutionary stream clustering algorithms. We list the limitations and the merits of each algorithm in the following:

- **evoStream**: **-Merits:** - Use idle times to improve the clustering quality - Output clusters at any time - Detection of outliers **-Limitations:** - Requires the set of the clusters number - Not suitable for high dimensional data.
- **DBStream**: **-Merits:** - Use the shared density between clusters to determine if two clusters can be merged - Robust to noise **-Limitations:** - Several parameters need to be set - Depends on the insertion order of the data points.
- **HDCStream**: **-Merits:** - Handles outliers - Improves the computation time and quality **-Limitations:** - Unable to detect variant levels of density - Can not handle high dimensional data.
- **FlockStream**: **-Limitations:** - Detects outliers - lower time complexity - Unable to handle high dimensional data.

4.3.7 GNG-based clustering algorithms

The Incremental Growing Neural Gaz [Prudent and Ennaji, 2005] algorithm has been proposed to follow the evolution of the graph. The method creates a new node each time the distance between the new data point and the existing node is higher than a threshold. This threshold value is a global parameter that corresponds to the average distance of the data to the center of the dataset. It has to be pre-defined, which makes it difficult to guess the right parameter to make the graph grow correctly. To resolve this weakness, **I2GNG** [Hamza et al., 2008] associates a threshold variable to each neuron. The only problem is that those thresholds have to be initialized at the beginning of the process.

AING [Bouguelia et al., 2013] is an incremental GNG that introduces an adaptive parameter-free distance threshold. It automatically learns the distance thresholds of nodes based on its neighbors and data points assigned to the node of interest. The algorithm overcomes the shortcoming

of excessive number of neurons by condensing them based on a probabilistic criterion and building a new topology with a fewer number of neurons, thus preserving time and memory resources.

Authors in [Ghesmoune et al., 2015] propose **G-Stream** based on growing neural gas. By introducing the notion of the reservoir to save distant points temporarily and applying a fading function, nodes can be created or removed during the learning process.

4.4 Comparison between Data Stream Clustering Algorithms

Table (4.1) and Table (4.2) list the differences between some clustering data stream methods. We presented two different comparisons. The first one compares the stream clustering algorithms based on their performances (scalability, the efficiency with high dimensional datasets, etc.). The second comparison is based on each algorithm's process, i.e., whether they remove, merge or split clusters, use fading or use a two-phase process.

Table 4.1 – *Comparison Between Data Stream Clustering Algorithms*

Algorithm	Method	Scalability	HDD[1]	Data Type	ABS[2]
SHC	Hierarchical	✗	✗	Numerical	✓
ODAC	Hierarchical	✓	✓	Time series	✓
ClusTree	Hierarchical	✓	✓	Numerical	✗
E-Stream	Hierarchical	✓	✓	Numerical	✓
HUE-Stream	Hierarchical	✓	✓	Mixed	✓
CluStream	Partitioning	✓	✗	Numerical	✗
RCD+	Partitioning	N/A*	✗	SQL Queries	✗
HP-Stream	Partitioning	✓	✗	Numerical	✗
SWClustering	Partitioning	N/A	N/A	Numerical	✗
Stream KM++	Partitioning	N/A	✓	Numerical	✗
StrAP	Partitioning	✓	N/A	Numerical	N/A
DCDGA	Density	✗	✗	Numerical	✓
CEC	Density	✗	✗	Numerical	✓
DenStream	Density	✓	✗	Numerical	✓
SDStream	Density	✓	✗	Numerical	✓
StreamOptics	Density	✓	✗	Numerical	✗
CEDGM	Grid	✗	✗	Numerical	✓
DGStream	Grid	N/A	N/A	Numerical	✓
ExCC	Grid	✓	✓	Mixed	✓
DGClust	Grid	✓	✓	Numerical	N/A
PKS-Stream	Grid	✓	✓	Numerical	✗
EDSSC	Subspace	✓	✓	Numerical	✗
CashStream	Subspace	✓	✓	Numerical	✓
SubClustTree	Subspace	✓	✓	Numerical	✗
HDDStream	Subspace	✓	✓	Numerical	✓
evoStream	Evolutionary	N/A	✗	Numerical	✓
HDCStream	Evolutionary	✓	✗	Numerical	✓
FlockStream	Evolutionary	✓	✗	Numerical	✓

[1] High Dimensional Data
[2] arbitrary-shaped clusters
* N/A: not available

Table 4.2 – *Comparison between the processes of each data stream clustering algorithm*

Algorithm	Phases	Remove	Merge	Split	Fade
SHC	online/offline	✓	✓	✓	✓
ODAC	online	✗	✓	✓	✗
ClusTree	online	✓	✓	✓	✓
E-Stream	online/offline	✓	✓	✓	✓
HUE-Stream	online/offline	✓	✓	✓	✓
CluStream	online/offline	✓	✓	✓	✓
RCD+	online	✗	✓	✓	✗
HP-Stream	online	✓	✗	✗	✓
SWClustering	online/offline	✓	✓	✓	✓
Stream KM++	online/offline	✓	✓	✓	✓
StrAP	online	✓	✓	✗	✗
DCDGA	online	✓	✗	✗	✓
CEC	online/offline	✓	✗	✗	✓
DenStream	online/offline	✓	✗	✗	✓
SDStream	online/offline	✓	✓	✗	✓
StreamOptics	online/offline	✓	✓	✗	✓
CEDGM	online	✓	✗	✗	✓
DGStream	online/offline	✓	✓	✓	✓
ExCC	online/offline	✓	✗	✗	✗
DGClust	online	✗	✓	✓	✓
PKS-Stream	online	✓	✗	✗	✓
EDSSC	online	✗	✗	✗	✗
CashStream	online	✗	✓	✓	✓
SubClustTree	online/offline	✓	✗	✗	✗
HDDStream	online/offline	✓	✓	✗	✓
evoStream	online/offline	✓	✓	✗	✓
HDCStream	online/offline	✓	✓	✗	✓
FlockStream	online	✓	✓	✗	✓

High Dimensional Data
Arbitrary-shaped clusters

4.5 Conclusion

In this chapter, we presented the concept of stream data, which can only be processed in one pass and must be analyzed following its order while respecting time and memory restrictions. We presented some processing techniques (One-pas, online/offline, time windows) and summarization (Random sampling, Sketching, Micro-clusters, Histograms, and Wavelets) of the stream data. We also defined the techniques of Dimensionality reduction and Load-shedding on stream data.

After presenting the techniques, we surveyed many representatives

and recent state-of-the-art algorithms for data stream clustering. These algorithms are categorized according to the nature of their underlying clustering approach, including hierarchical, partitioning, density, grid-based stream methods, Evolutionary and subspace clustering algorithms. We performed a detailed comparison of these algorithms.

Part II

Clustering Data Stream Approaches

5 Subspace Data Stream Clustering with Global and Local Weighting Models

5.1 Introduction

Subspace clustering discovers clusters embedded in multiple, overlapping subspaces of high dimensional data. It has been successfully applied in many domains. Data streams are ordered and potentially infinite sequences of data points created by a typically non-stationary data generating process. Clustering this type of data requires some restrictions in time and memory. In this section, we propose the S2G-Stream algorithm based on growing neural gas and soft subspace clustering. We introduce two types of entropy weighting for both features and blocks, and also two weighting models (local and global). Experiments on public datasets demonstrated the ability of S2G-Stream to detect relevant features and blocks and to provide the best partitioning of the data.

5.2 Proposed Method

In this section we introduce S2G-Stream based on the Growing Neural Gas (GNG) model taking into account the block structuring of features. We assume that features and blocks of features contribute at different levels to the determination of clusters. These contributions made by the features and blocks in each class are then measured by weights. We assume that the data stream consists in a sequence $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ of n (potentially infinite) elements arriving at times $t_1, t_2, ..., t_n$, where $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$. At each time, S2G-Stream is represented by a graph $\mathcal{C}$ with K nodes, where each node represents a cluster. Each node $c \in \mathcal{C}$ is associated with: (1) A prototype $\mathbf{w}_c = (w_c^1, w_c^2, \ldots, w_c^d)$ representing its position (2) A weight π_c (3) An error variable $error(c)$ representing the distance between this node and the assigned data points. For each pair of nodes (r, c), we denote the shortest path linking r and c by $\delta(c, r)$ the length of the shortest chain linking r and c on the graph. $\mathcal{K}^T(\delta) = \mathcal{K}(\delta/T)$ is the neighborhood function, T controls the width of $\mathcal{K}$.

Based on [Ouattara et al., 2013, Chen et al., 2012], S2G-Stream introduces a double weighting system for features denoted by β and subspaces denoted by α. From the two types of weights, the subspaces of clusters

can be revealed. We propose two types of weighting: the local weighting model (LWM), where the clusters influence the weights, and each feature and block have different weight vector for each cluster. And the global weighting model (GWM) where the weights are independent of the clusters, each feature and block have the same weight vector. Table (5.1) describes notations used in our method S2G-Stream.

Notation	Description
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$	n Data Stream
$\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$.	d-dimensional data point
$\mathbf{w}_c = (w_c^1, w_c^2, \ldots, w_c^d)$	Prototype of node c
π_c	Weight of node c
$error(c)$	Accumulated error for node c
$\delta(c, r)$	Shortest path linking c and r
P	Number of blocks
α	Matrix of block weights
β	Matrix of feature weights
λ and η	Adjustment parameters for feature and block weights
γ	Decay factor
π_{min}	the minimum weight of a node
τ_{age}	edge age growth rate
age_{max}	maximum age of an edge
μ	the number of nodes to add
bmu_1 and bmu_2	best matching units (the nearest and second nearest nodes)
$\phi(\mathbf{x}_i)$	assignment function (cluster corresponding to data point) $\mathbf{x_i}$

Table 5.1 – *Notations used in S2G-Stream*

Main contribution

In the previous work [Ghesmoune et al., 2015], authors consider all features equally important to the clustering task. However, some features or subspaces of features might be more influential to the clustering process. Our contribution in this chapter is to introduce a double weight system to make relevant features and subspaces contribute more to the clustering process. The contribution is made by introducing the weights into the cost function of our algorithm. In our previous works, [Attaoui et al., 2019b;a], we proposed a local weighting model that depends on each cluster. The weights are different from one cluster to another, which makes it difficult to obtain scores to compare subspaces and features. Therefore, further experiments could not be conducted. The method presented in this chapter has the following merits compared to the previous works:

- The introduction of the Global and Local Weighting Model. We used these weights as scores to conduct more experiments. The subspace clustering method with the Global Weighting Model was used as a dimensionality reduction method.

- For the stream subspace clustering, more tests were conducted in this chapter to respond to the following questions: Does the order of data significantly impact the data stream analyzes? Does the overlap between data windows impact the data stream analyzes? Is the type of weighting model important for the clustering process?

Cost function

Based on [Chen et al., 2012] and [Ouattara et al., 2013], we propose to minimize the new cost function defined below for data batch $\mathcal{X}^{(t+1)} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{t+1}\}$ with two weighting models. Figure 5.1 illustrates the difference between the two weighting models presented below.

Local Weighting Model (LWM)

In local weighting model, α is a $K \times P$ matrix where α_c^b is the weight of block b in node c of $\mathcal{X}$. β is a $K \times d$ matrix where β_b is a $K \times d_b$ matrix, where $\beta_{cb}^j (j = 1, ..., d_b)$ is the weight of the j^{th} feature in block b for node c with $\sum_{j=1}^{d_b} \beta_{cb}^j = 1$ and $\sum_{b=1}^{P} \alpha_c^b = 1, \forall c \in \mathcal{C}$.

$$J_{(LWM)}^{(t+1)}(\phi, \mathcal{W}, \alpha, \beta) = \sum_{c \in C} \sum_{b \in P} \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} \mathcal{K}^T(\delta(c, \phi(\mathbf{x}_i)))\alpha_c^b \mathcal{D}_{\beta_{cb}} + J_{cb} + I_c \tag{5.1}$$

Where:

$$I_c = \lambda \sum_{b=1}^{P} \alpha_c^b \log(\alpha_c^b)$$

$$J_{cb} = \eta \sum_{j=1}^{d_b} \beta_c^j \log(\beta_c^j)$$

And:

$$\mathcal{D}_{\beta_{cb}} = \sum_{j=1}^{d_b} \beta_c^j (x_i^j - \omega_c^j)^2$$

I_c and J_{cb} respectively represent the weighted negative entropies associated with the block weight vectors and the feature weight vectors. The parameters λ and η are used to adjust the relative contributions made by the features and blocks to the clustering.

Global Weighting model (GWM)

In order to see if the weights can be meaningful and show the importance of blocks and features when these weights are independent from prototypes $\mathcal{W}$, we present in new cost function eq. (5.2) another cost function where the weights α and β are global and does not depend on prototypes $\mathcal{W}$.

For this model, α is a $1 \times P$ matrix where α_b is the weight of block b. β is a $1 \times d$ matrix where β_b is a $1 \times d_b$ matrix, where $\beta_b^j (j = 1, ..., d_b)$ is the weight of the j^{th} feature in block b with $\sum_{j=1}^{d_b} \beta_b^j = 1$ and $\sum_{b=1}^{P} \alpha_b = 1$.

$$J^{(t+1)}_{(GWM)}(\phi, \mathcal{W}, \alpha, \beta) = \sum_{c \in C} \sum_{b=1}^{P} \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} \mathcal{K}^T(\delta(c, \phi(\mathbf{x}_i)))\alpha_b \mathcal{D}_{\beta_b} + J_b + I \quad (5.2)$$

Where

$$I = \lambda \sum_{b=1}^{P} \alpha_b \log(\alpha_b)$$

$$J_b = \eta \sum_{j=1}^{d_b} \beta^j \log(\beta^j)$$

And

$$\mathcal{D}_{\beta_b} = \sum_{j=1}^{d_b} \beta^j (x_i^j - \omega_c^j)^2$$

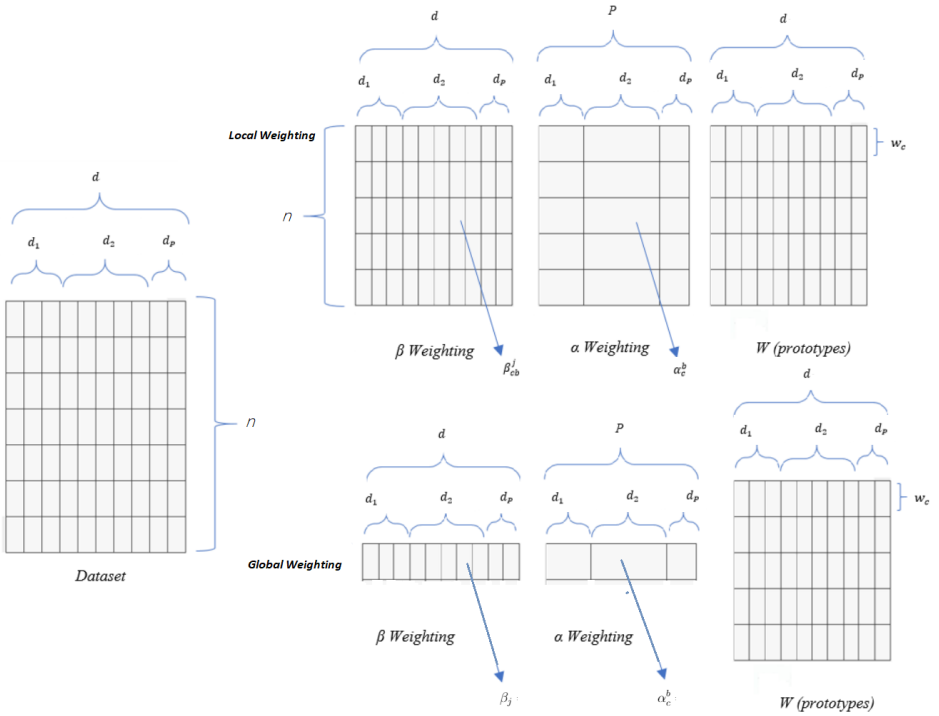


Figure 5.1 – *Difference between Global and Local Weighting models.* α_c^b *is the weight of the subspace b in the node c and* $\beta_{cb}^j (j = 1, ..., d_b)$ *is the weight of the* j^{th} *feature in the subspace b for the node c.* w_c *is the prototype of node c.*

Optimization Algorithm

The optimization of the cost function is performed alternately for each batch $\mathcal{X}^{(t+1)}$ in four steps corresponding to the four parameters $\mathcal{W}, \phi, \alpha$ and β:

1. **Assignment function:** For a fixed $\mathcal{W}, \alpha$ and β, the assignment function $\phi(\mathbf{x}_i)$ is described below for both local and global weighting models. In order to reduce the computational cost, neighboring nodes are not considered in the assignment.

- Local Weighting model

The assignment function for LWM is described in Equation (5.3)

$$\phi(\mathbf{x}_i) = \arg\min_{c \in \mathcal{C}} \left(\sum_{b=1}^{P} \alpha_c^b \sum_{j=1}^{d_b} \beta_c^j \left(x_i^j - \omega_c^j \right)^2 \right) \tag{5.3}$$

- Global Weighting model

The assignment function for GWM is described in Equation (5.4)

$$\phi(\mathbf{x}_i) = \arg\min_{c \in \mathcal{C}} \left(\sum_{b=1}^{P} \alpha_b \sum_{j=1}^{d_b} \beta^j \left(x_i^j - \omega_c^j \right)^2 \right) \tag{5.4}$$

2. **Update prototypes** $\mathcal{W}$**:** For a fixed ϕ, α and β the prototypes $\mathbf{w}_c$ are updated for every batch of data following the equation defined below. Since the prototypes are independent of the weights, we have only one update function for both models.

$$\mathbf{w}_c^{(t+1)} = \frac{\mathbf{w}_c^{(t)} n_c^{(t)} \gamma + \sum_{r \in C} \mathcal{K}^T(\delta(r,c)) \mathbf{w}_r^{(t)} m_r^{(t)}}{n_c^{(t)} \gamma + \sum_{r \in C} \mathcal{K}^T(\delta(r,c)) m_r^{(t)}} \tag{5.5}$$

where $\mathbf{w}_c^{(t)}$ is the previous prototype, $n_c^{(t)}$ is the number of points assigned to the cluster, $\mathbf{w}_r^{(t)}$ is the previous prototype for the cluster r (which is a neighbor of c) and $m_r^{(t)}$ is the number of points added to the cluster r in the current batch: $n_c^{(t+1)} = n_c^{(t)} + m_c^{(t)}$.

3. **Update weights** α for a fixed ϕ, $\mathcal{W}$ and β, we minimize the objective function (5.1) with respect to α_c^b the weight of block b in the c-th cluster. Since there exists a constraint $\sum_{b=1}^{P} \alpha_c^b = 1$. We form the Lagrangian by isolating the terms which contain α and adding Lagrangian multipliers μ as follows:

$$\mathcal{L}_{LWM}(\alpha, \lambda) = J^{(t+1)}(\phi, \mathcal{W}, \alpha, \beta) - \sum_{b \in P} \mu_c \left(\sum_{b=1}^{P} \alpha_{cb} - 1 \right) \tag{5.6}$$

Taking the derivative with respect to α_c^b and setting it to zero yields a minimum of α_c^b as follows.

- Local Weighting model

$$\alpha_c^b = \frac{e^{\frac{-D_{cb}}{\lambda}}}{\sum_{s=1}^{P} e^{\frac{-D_{cs}}{\lambda}}} \tag{5.7}$$

with

$$D_{cb} = \sum_{\mathbf{x}_i \in \mathcal{X}^{(t)}} \mathcal{K}^T(\delta(\phi(\mathbf{x}_i), c)) \sum_{j=1}^{d_b} \beta_c^j \left(\mathbf{x}_i^j - w_c^j \right)^2 \tag{5.8}$$

- Global Weighting model

$$\alpha_c^b = \frac{e^{\frac{-D_b}{\lambda}}}{\sum_{s=1}^{P} e^{\frac{-D_s}{\lambda}}} \tag{5.9}$$

with

$$D_b = \sum_{\mathbf{x}_i \in \mathcal{X}^{(t)}} \mathcal{K}^T(\delta(\phi(\mathbf{x}_i), c)) \sum_{j=1}^{d_b} \beta^j \left(\mathbf{x}_i^j - w_c^j\right)^2 \tag{5.10}$$

4. **Update weights** β**:** for a fixed ϕ, $\mathcal{W}$ and β, we minimize the objective function (5.1) with respect to β_{cb}^j (the weight of feature j of block b in the $c-th$ cluster). Since there exist a constraint $\sum_{j=1}^{d_b} \beta_{cb}^j = 1$. We form the Lagrangian by isolating the terms which contain β and adding Lagrangian multipliers μ as follows:

$$\mathcal{L}_{LWM}(\alpha, \lambda) = J^{(t+1)}(\phi, \mathcal{W}, \alpha, \beta) - \sum_{c \in C} \mu_{cb} (\sum_{j=1}^{d_b} \beta_{cb}^j - 1) \tag{5.11}$$

Taking the derivative with respect to β_{cb}^j and setting it to zero yields a minimum of β_{cb}^j as follow:

- Local Weighting model

The update function for LWM is presented in Equation (5.12)

$$\beta_c^j = \frac{e^{\frac{-E_c^j}{\eta}}}{\sum_{h \in P_j} e^{\frac{-E_c^h}{\eta}}} \tag{5.12}$$

with

$$E_c^j = \sum_{\mathbf{x}_i \in \mathcal{X}^{(t)}} \alpha_c^{b_j} \mathcal{K}^T(\delta(\phi(\mathbf{x}_i), c)) \left(\mathbf{x}_i^j - w_c^j\right)^2 \tag{5.13}$$

where b_j is the block where the j^{th} feature belongs.

- Global Weighting model

The update function for GWM is presented in Equation (5.14):

$$\beta_j = \frac{e^{\frac{-E_j}{\eta}}}{\sum_{h \in P_j} e^{\frac{-E_h}{\eta}}} \tag{5.14}$$

with

$$E_j = \sum_{\mathbf{x}_i \in \mathcal{X}^{(t)}} \alpha_{b_j} \mathcal{K}^T(\delta(\phi(\mathbf{x}_i), c)) \left(\mathbf{x}_i^j - w_c^j\right)^2 \tag{5.15}$$

S2G-Stream algorithm

S2G-Stream aims at extending the G-Stream algorithm [Ghesmoune et al., 2016a] to subspace clustering by introducing block and feature entropy weighting. Starting with two nodes, and as a new data point is available, we link the nearest and the second-nearest nodes by an edge. The nearest node with its topological neighbors is moved towards the data point. We present below the main functions of the S2G-Stream algorithm.

Fading function

Most data stream algorithms consider most recent data as more important and reflect better the changes in the data distribution. For that, the notion of time windows is used. There are three window models commonly studied in data streams: landmark, sliding and damped [Zhu and Shasha, 2002]. We consider the damped window model, in which the weight of each node decreases exponentially with time via a *fading* function by introducing a decay factor parameter $0 < \gamma < 1$.

$$\pi_c^{(t+1)} = \pi_c^{(t)} \gamma \tag{5.16}$$

If the weight of a node is less than a threshold value, this node is considered outdated and removed (along with its links).

Edge management

An edge linking two nodes can be strengthened or removed. Its age grows with the exponential function $2^{\tau_{age}^{(t-t_0)}}$, where $\tau_{age} > 0$ defines growth rate of the age over time, t denotes the current time and t_0 is the creation time of the edge. A new edge can be added to connect two nodes. It can be removed if it exceeds the maximum age.

Algorithm 8 Edge Management

-Increment the age of all edges emanating from bmu and weight them;
-Create an edge between bmu_1 and bmu_2. If it already exists:
set its age to zero ;
-Remove the edges whose age is greater than age_{max};

Node insertion

Nodes can be inserted into the graph between the two nodes having the highest error value. If the weight of a node is lower than a threshold value, then this node is considered as outdated and removed (along with its links). Figure 5.2 illustrates the process of insertion of 3 nodes. The description of node Insertion process can be found in Algorithm (9)

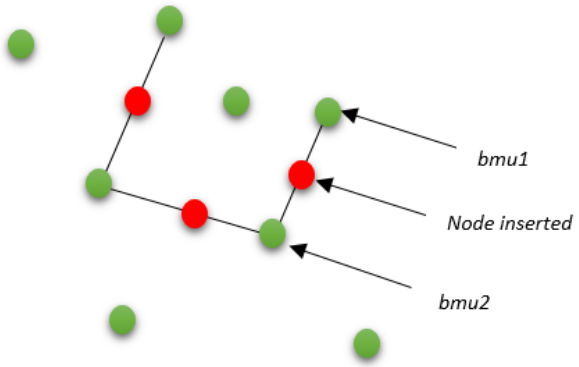


Figure 5.2 – *An insertion of 3 nodes during the same time window.*

Algorithm 9 Node Insertion

-Find the node q with the largest error and its neighbor f with the largest accumulated error;
-Add the new node r between nodes q and f: $\mathbf{w}_r = 0.5(\mathbf{w}_q + \mathbf{w}_f)$;
-Decrease the error variables of q and f by multiplying them by a constant v where $0 < v < 1$ and assign to r the error value of q ;
-Decrease the error of all nodes by multiplying them by a constant s, and remove isolated nodes ;

The complete description of the S2G-Stream algorithm can be found in Algorithm (10).

Algorithm 10 S2G-Stream

input : $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}, \pi_{min}, \tau_{age}, age_{max}, d, \eta, \lambda, \mu, \gamma, P$
output : prototypes: $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, .., \mathbf{w}_n\}$, feature weights matrix β and subspace weights matrix α

Initialize the graph with two nodes, initialize α and β weights randomly;
while *there is a micro-batch to proceed* **do**
  -Get the micro-batch of data points arrived at time interval t ;
  for ***each*** *data-point in the current micro-batch* **do**
    Assignment Step
    -Find the nearest and second nearest nodes bmu_1 and bmu_2 ;
    -Assign each point to the closest center (bmu_1) following Equation (5.3) for LWM and Equation (5.4) for GWM;
    Update Step
    -Update the new centroid as described in Equation (5.5) ;
    Edge Management following Algorithm (8)
    -Update the *error* of each node: $error(bmu_1) = error(bmu_1) + ||x_i - bmu_1||^2$;
    Fading Function

    Add Nodes following Algorithm (9)

    Update weights
    -LWM model: update feature weights α following Equation (5.7) and block weights β following Equation (5.12);
    -GWM model: update feature weights α following Equation (5.9) and block weights β following Equation (5.14);

5.3 Experimental Results

Datasets and Quality Criteria

The S2G-Stream method described in this article was implemented in Spark/Scala and is available on Clustering4Ever github repository[1]. We evaluated clustering quality of S2G-Stream on several synthetic and real dataset. Synthetic datasets are $DS1$ and $DS2$ generated using this tools [2]. The real datasets were taken from the UCI repository [Frank and Asuncion, 2010] and are described below:

- *Waveform*: Each class is generated from a combination of 2 of 3 "base" waves, the second 20 features contains noise (mean 0, variance 1).
- *Image Segmentation*(IS): Image data described by high-level numeric-valued attributes. The instances were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel. Each instance is a 3x3 region.
- *Cardiotocography*(CTG): 2126 fetal cardiotocograms(CTGs) were automatically processed and the respective diagnostic features measured. The CTGs were also classified by three expert obstetricians and a consensus classification label assigned to each of them.

[1] https://github.com/Clustering4Ever/Clustering4Ever
[2] http://impca.curtin.edu.au/local/software/synthetic -data-sets.tar.bz2

- *pendigits*: Pen-based recognition of handwritten digits data, set-digit database of 250 samples from 44 writers.

Dataset	Number of Features	Number of Blocks	Number of Instances
Waveform	40	2 (20,20)	5000
CTG	21	3 (7,4,10)	2126
IS	19	2 (9,10)	2310
pendigits	17	2 (10,7)	10992
DS1	2	/	9153
DS2	2	/	5458

Table 5.2 – *Description of datasets used in experimentation*

For the quality measures, we used Normalized Mutual Information (NMI) [Strehl and Ghosh, 2002] and Adjusted Rand index (ARAND) [Hubert and Arabie, 1985] described in Section 3.2.2.

Experimental Settings

Assuming large high-dimensional data arrives as a continuous stream, S2G-Stream divides the streaming data into batches and processes each batch continuously. The batch size depends on the available memory and the size of the original dataset. We set the time interval between two batches to 1 second. The parameters of S2G-Stream are described in table (5.1). We repeated our experiments with different initialization and have chosen those giving the best results. We set $\mu = 3$, $\gamma = 0.99$ and $age_{max} = 250$. λ and η and the batch size for each dataset are described in Table (5.3). The weights α and β are initialized randomly under the two constraints $\sum_{j=1}^{d_b} \beta_{cb}^{j} = 1$ and $\sum_{b=1}^{P} \alpha_{c}^{b} = 1, \forall c \in \mathcal{C}$.

Datasets	λ	η	$\lvert Batch \rvert$
Waveform	5	15	100
IS	3	31	100
CTG	7	11	100
pendigits	3	17	1000
DS1	7	11	300
DS2	7	11	300

Table 5.3 – *Initialization of parameters λ and η and batch size for each dataset*

Clustering Evaluation

To show the effectiveness of our method, we compared it to different subspace and stream clustering algorithms. As stream clustering methods we chose *CluStream* and *DStream* from R package streamMOA[3]. For subspace clustering methods, we implemented *2S-SOM* [Ouattara et al., 2013] in the Scala language, and the code is available on the C4E GitHub repository. We compared the method to CLIQUE[4], PROCLUS[5], and W-K-

[3] https://github.com/mhahsler/streamMOA
[4] https://github.com/georgekatona/Clique
[5] https://github.com/OguzhanOktay-Buyuk/PROCLUS-Python3

means[6]. We also compared our method to the GNG algorithm. For each algorithm, we repeated the experiments 10 times on each dataset. The results are reported in Table (5.4) and Table (5.5). It is noticeable that S2G-Stream gives better results than the other methods except for DStream on *CTG* and *Waveform* with NMI metric and on *CTG* and *DS*1 with ARAND metric.

We observe that the global weighting model of S2G-Stream(GWM) performs better than the Clustream algorithm. But DStream gives better results in most cases. These results are since S2G-Stream detects noisy features, which allows relevant features to contribute more to clustering. The notion of fading also improves the clustering quality of our method compared to the other methods by reducing the impact of irrelevant data. For the subspace clustering methods, our method outperforms most of the methods except for CLIQUE on the *waveform* dataset with ARAND metric, and W-K-means on *CTG* with ARAND metric. We recall that all the subspace algorithms and GNG make several iterations on data while all our algorithm makes just one pass over the data. The four subspace clustering algorithms provide good detection of relevant subspace, but the clustering process is unable to detect noise and arbitrary shaped clusters. Our method detects both relevant subspaces and features. Those detected subspaces contribute to the clustering process to improve the clustering quality.

Dataset	Metrics	S2G-Stream (LWM)	S2G-Stream (GWM)	GNG	CluStream	DStream
waveform	NMI	0.397±0.002	0.355±0.018	0.306±0.078	0.393±0.065	**0.434±0.003**
	ARAND	0.137±0.007	**0.339±0.021**	0.006±0.103	0.010±0.001	0.040±0.001
IS	NMI	**0.550±0.05**	0.364±0.039	0.542±0.010	0.506±0.065	0.435±0.07
	ARAND	**0.418±0.04**	0.183±0.006	0.102±0.051	0.098±0.010	0.134±0.002
CTG	NMI	0.270±0.009	0.237±0.023	0.375±0.004	0.086±0.06	**0.471±0.170**
	ARAND	0.124±0.005	0.092±0.018	0.030±0.011	0.019±0.008	**0.209±0.002**
pendigits	NMI	**0.672±0,038**	0.362±0.011	0.585±0.019	0.285±0.099	0.554±0.15
	ARAND	**0.408±0.060**	0.128±0.007	0.027±0.085	0.011±0.006	0.016±0.011
DS1	NMI	**0.737±0.222**	0,369±0.008	0.622±0.037	0.643±0.029	0.639±0.009
	ARAND	0.299±0.106	0,114±0.008	0.042±0.032	0.109±0.031	**0.406±0.006**
DS2	NMI	**0.677±0.111**	0,286±0,010	0.640±0.063	0.334±0.113	0.483±0.017
	ARAND	**0.303±0.071**	0,097±0,004	0.063±0.033	0.127±0.088	0.060±0.019

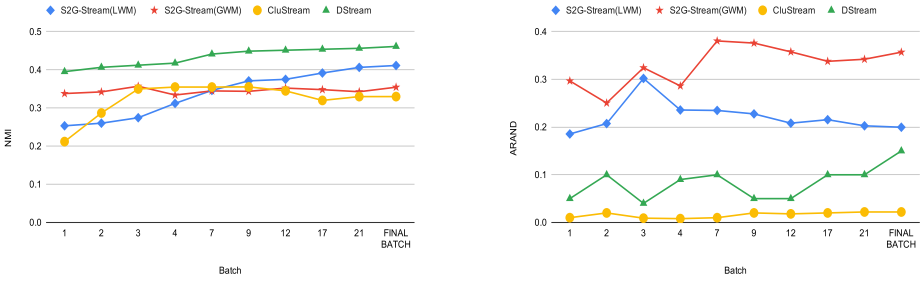
Table 5.4 – *Comparing S2G-Stream Local (LWM) and Global Weighting Model (GWM) with different stream algorithms and GNG. The first value is the average of 10 repetitions followed by the standard deviation.*

[6]https://github.com/Yanis2016/Weighted-K-Means-clustering

Dataset	Metrics	S2G-Stream (LWM)	S2G-Stream (GWM)	2S-SOM	CLIQUE	PROCLUS	W-K-means
waveform	NMI	**0.397 ± 0.002**	0.355 ± 0.018	0.060 ± 0.01	0.381 ± 0.5	0.17 ± 0.04	0.335 ± 0.07
	ARAND	0.137 ± 0.007	0.339 ± 0.021	0.097 ± 0.01	**0.345 ± 0.3**	0.163 ± 0.03	0.313 ± 0.02
IS	NMI	**0.550 ± 0.05**	0.364 ± 0.039	0.091 ± 0.06	0.372 ± 0.04	0.120 ± 0.13	0.228 ± 0.09
	ARAND	**0.418 ± 0.04**	0.183 ± 0.006	0.023 ± 0.01	0.104 ± 0.00	0.058 ± 0.07	0.228 ± 0.08
CTG	NMI	**0.270 ± 0.01**	0.237 ± 0.023	0.200 ± 0.07	0.024 ± 0.00	0.056 ± 0.02	0.192 ± 0.04
	ARAND	0.124 ± 0.005	0.092 ± 0.018	0.017 ± 0.08	0.008 ± 0.05	0.036 ± 0.01	**0.132 ± 0.00**
pendigits	NMI	**0.672 ± 0.038**	0.362 ± 0.011	0.033 ± 0.03	0.255 ± 0.00	0.367 ± 0.43	0.222 ± 0.05
	ARAND	**0.408 ± 0.060**	0.128 ± 0.007	0.017 ± 0.06	0.002 ± 0.00	0.305 ± 0.40	0.162 ± 0.04
DS1	NMI	**0.737 ± 0.222**	0.369 ± 0.008	0.210 ± 0.00	0.387 ± 0.08	0.315 ± 0.09	0.418 ± 0.19
	ARAND	**0.299 ±0 .106**	0.114 ± 0.008	0.087 ± 0.01	0.147 ± 0.09	0.119 ± 0.05	0.202 ± 0.09
DS2	NMI	**0.677 ± 0.111**	0.286 ± 0.010	0.127 ± 0.02	0.301 ± 0.02	0.379 ± 0.22	0.256 ± 0.11
	ARAND	**0.303 ± 0.071**	0.097 ± 0.004	0.019 ± 0.00	0.124 ± 0.02	0.228 ± 0.17	0.229 ± 0.04

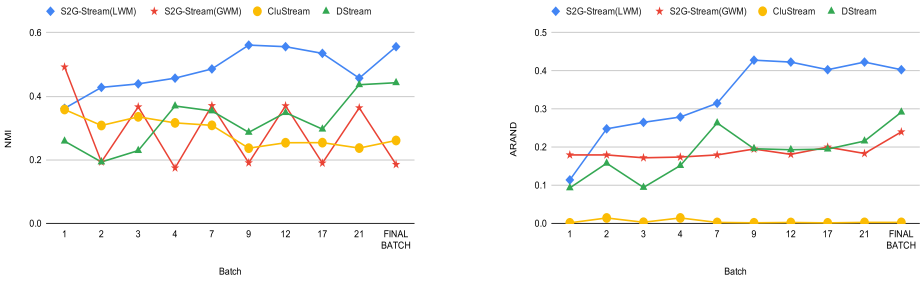
Table 5.5 – *Comparing S2G-Stream Local (LWM) and Global Weighting Model (GWM)) with different subspace algorithms. The first value is the average of 10 repetitions followed by the standard deviation.*

Figures (5.3), (5.4), (5.5), (5.6), (5.7) and (5.8) show a comparison of our models to *CluStream* and *DStream* algorithms in terms of NMI and ARAND for each window. For almost all cases, the NMI and ARAND for S2G-Stream(LWM) are higher than for *CluStream* and *DStream*, except for some windows and also some datasets (*DStream* on waveform with NMI). This is due to the evolution of the nodes with S2G-Stream, and its capacity to remove noisy points and outdated prototypes over time, while the number of nodes of *CluStream* and *DStream* remains static. The results of S2G-Stream(GWM) are better than those of CluStream on most windows. DStream algorithm outperforms our second model (in certain cases). S2G-Stream shows a greater ability to partition high-dimensional data and is more stable in subspace clustering analysis.



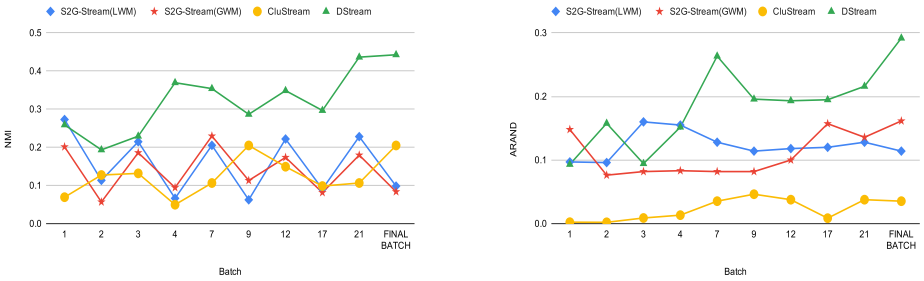
(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.3 – *Evolution of NMI and ARAND for Waveform dataset compared with CluStream and DStream algorithms.*



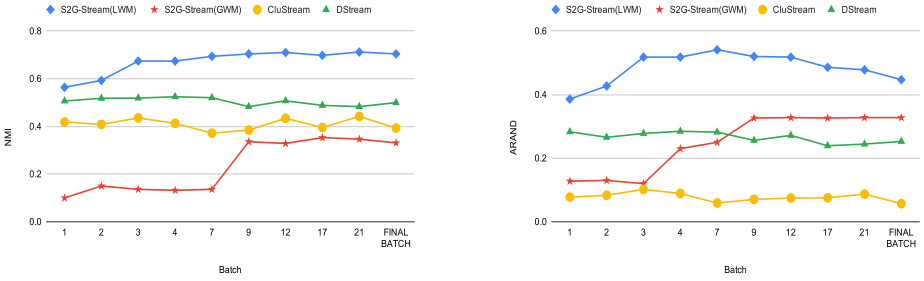
(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.4 – *Evolution of NMI and ARAND for IS dataset compared with CluStream and DStream algorithms.*



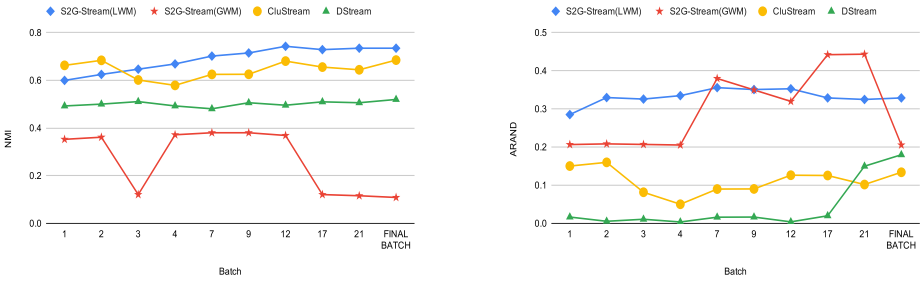
(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.5 – *Evolution of NMI and ARAND for CTG dataset compared with CluStream and DStream algorithms.*



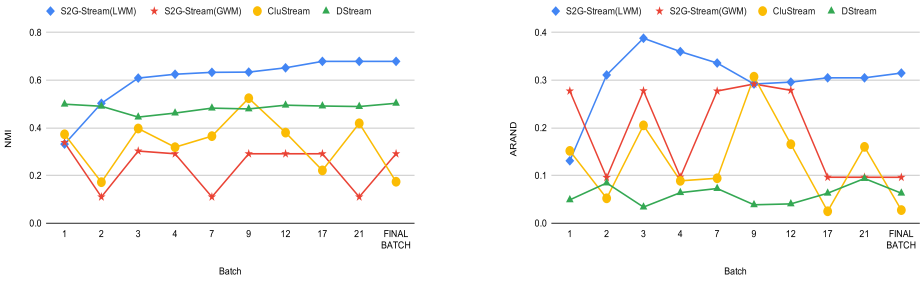
(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.6 – *Evolution of NMI and ARAND for pendigits dataset compared with CluStream and DStream algorithms.*



(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.7 – *Evolution of NMI and ARAND for DS1 dataset compared with CluStream and DStream algorithms.*



(a) *NMI Evolution* (b) *ARAND Evolution*

Figure 5.8 – *Evolution of NMI and ARAND for DS2 dataset compared with CluStream and DStream algorithms.*

Detection of subspaces

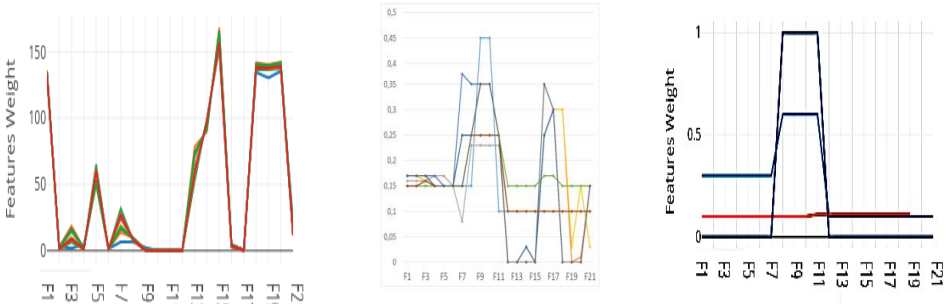
For this section, we settled to two datasets *Waveform* and *CTG* since the blocks of this datasets are defined in their descriptions. *CTG* dataset describes fetal cardiotocograms and is composed of 3 blocks. Block 1 contains seven features related to the heart rate of a fetus. Block 2 contains four features describing heart rate variability. Block 3 is composed of 10 features defining histograms of fetal cardiography. *Waveform* dataset is composed of 2 blocks of 20 features, where the second block is composed of noisy features.

Local Weighting model

Figure (5.9) represents prototypes $\mathcal{W}$, β weights and α weights for the final batch of CTG dataset. We observe in Figure (5.9b) that weights of features (8,9,10,11) which are respectively ASTV[7], MSTV[8], ALTV[9] and MLTV[10], are higher than the weights of the other features for most clusters. We observe that these 4 features influence better the clustering process and are more important than the other features for most clusters. In Figure (5.9c), we observe that weight α of the second block that contains these four features

[7] percentage of time with abnormal short term variability
[8] mean value of short term variability
[9] percentage of time with abnormal long term variability
[10] mean value of long term variability

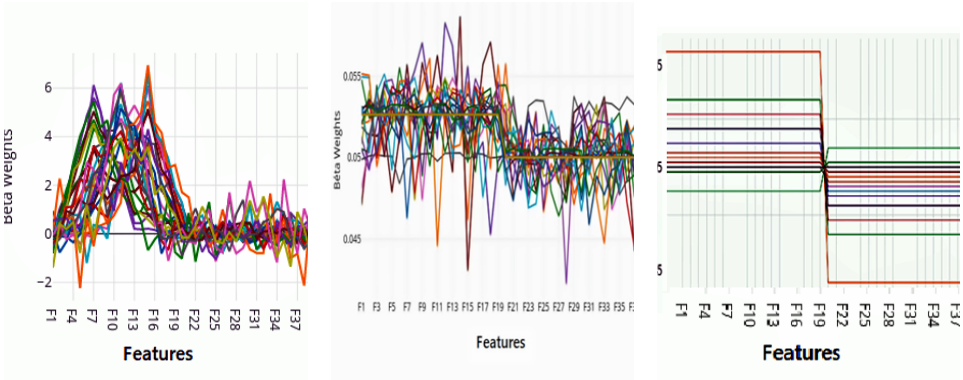
is also higher than the weights of the other two blocks. We conclude from this experiment that heart rate variability influences the clustering of fetal cardiotocograms.



(a) *Prototypes* $\mathcal{W}$ (b) β *weights* (c) α *weights*

Figure 5.9 – *Results of local weights* α *and* β*, and prototypes* $\mathcal{W}$ *for the final batch for CTG dataset. Each color represent a node.*

Figure (5.10) illustrates the capability of S2G-Stream to detect noise on the *Waveform* dataset. It represents prototypes $\mathcal{W}$, weights β, and weights α for the final batch of *Waveform* dataset. We can clearly observe in Figure (5.10b) that weights of the first 20 features increase over time, while weights of the 20 other features decrease. The weights of the 20 noisy features are much lower than the weights of the other features. In Figure (5.10c), weights α for block 1 are higher than the weights of block two, which make sense since the second block contains only noisy features. We can see that our algorithm assigns higher weights to the first 20 features, while the weights of noisy features are lower.



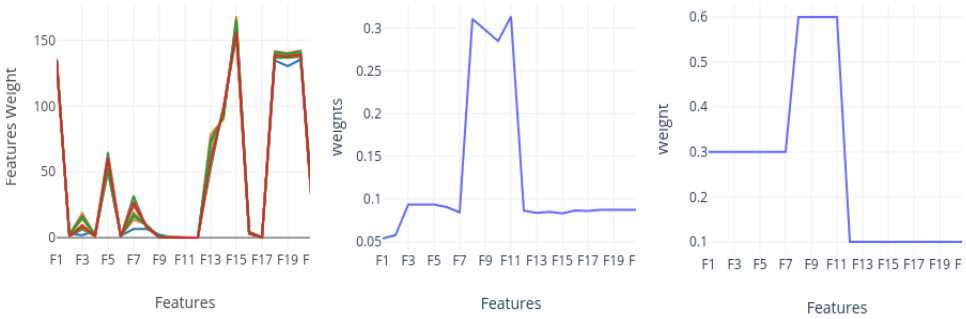
(a) *Prototypes* $\mathcal{W}$ (b) β *weights* (c) α *weights*

Figure 5.10 – *Results of local weights* α *and* β*, and prototypes* $\mathcal{W}$ *for the final batch for Waveform dataset. Every color represent a node.*

Global Weighting model

To show the effectiveness of the global weighting model, we report in Figure (5.11) and Figure (5.12) the weights α, β and prototypes $\mathcal{W}$ for the final batch for *CTG* and *Waveform* datasets, for multiple training epochs.

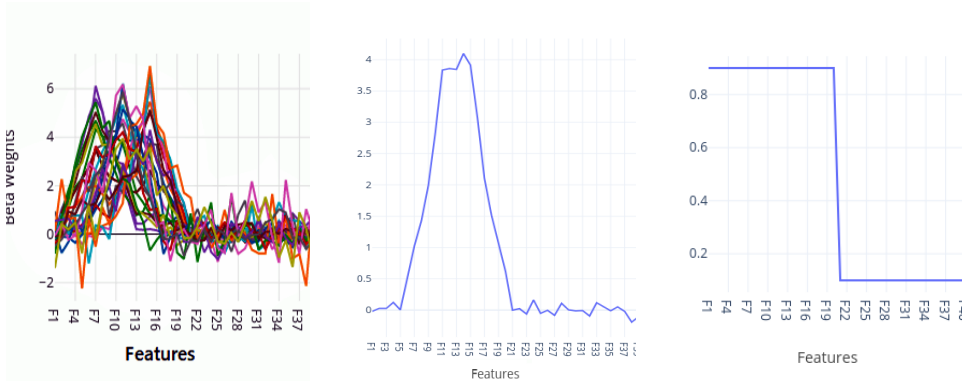
For *CTG* dataset, the blocks are detected more accurately than by the local weighting model. The weight of second block and the features contained within it is always higher than the weights of the other blocks and their features. Prototypes $\mathcal{W}$ is the same as in the first model since the prototypes update function is the same for both models.



(a) *Prototypes* $\mathcal{W}$ (b) β *weights* (c) α *weights*

Figure 5.11 – *Results of global weights* α *and* β*, and prototypes* $\mathcal{W}$ *for the final batch for CTG dataset for the second weighting model.*

For *Waveform* datasets, better blocks are also detected, but the model assigns lower weights to some relevant features detected by LWM (features 1 to 5). This is due to the fact that the model does not take into consideration the nodes and their preferences. We can assume that the first model is well adapted for this kind of datasets.



(a) *Prototypes* $\mathcal{W}$ (b) β *weights* (c) α *weights*

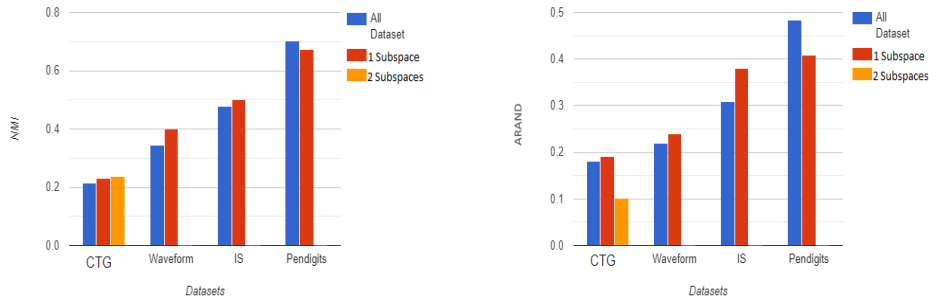
Figure 5.12 – *Results of global weights* α *and* β*, and prototypes* $\mathcal{W}$ *for the final batch for Waveform dataset for the second weighting model.*

Feature Selection with GWM

Even if the results of GWM are lower than the results of LWM, as we see in Table (5.4), GWM gave us a global weighting that we will use further to select the best features. We can see here the importance of the GWM on subspace detection.

Based on the previous experiments, we performed S2G-Stream on only the reduced dataset i.e., only the relevant blocks from the dataset based on

the average of block weights α. We present the NMI and ARAND results compared with the results on the whole dataset in Figure 5.13. For the *CTG* dataset, both NMI and ARAND results on the reduced dataset are better than the results using all features and blocks. Block 1 with block 2 provides a lower ARAND value. We conclude that block 1 gives better results compared to other blocks, which confirms previous results. For the *Waveform* dataset, NMI and ARAND measured on the reduced dataset (block 1) are higher than the whole dataset since the noisy features on the whole dataset affect the results. The same results are observed on the *IS* dataset. For *pendigits*, the results of NMI and ARAND on the reduced dataset are lower, since both blocks contain features that are required for the pen-based recognition.

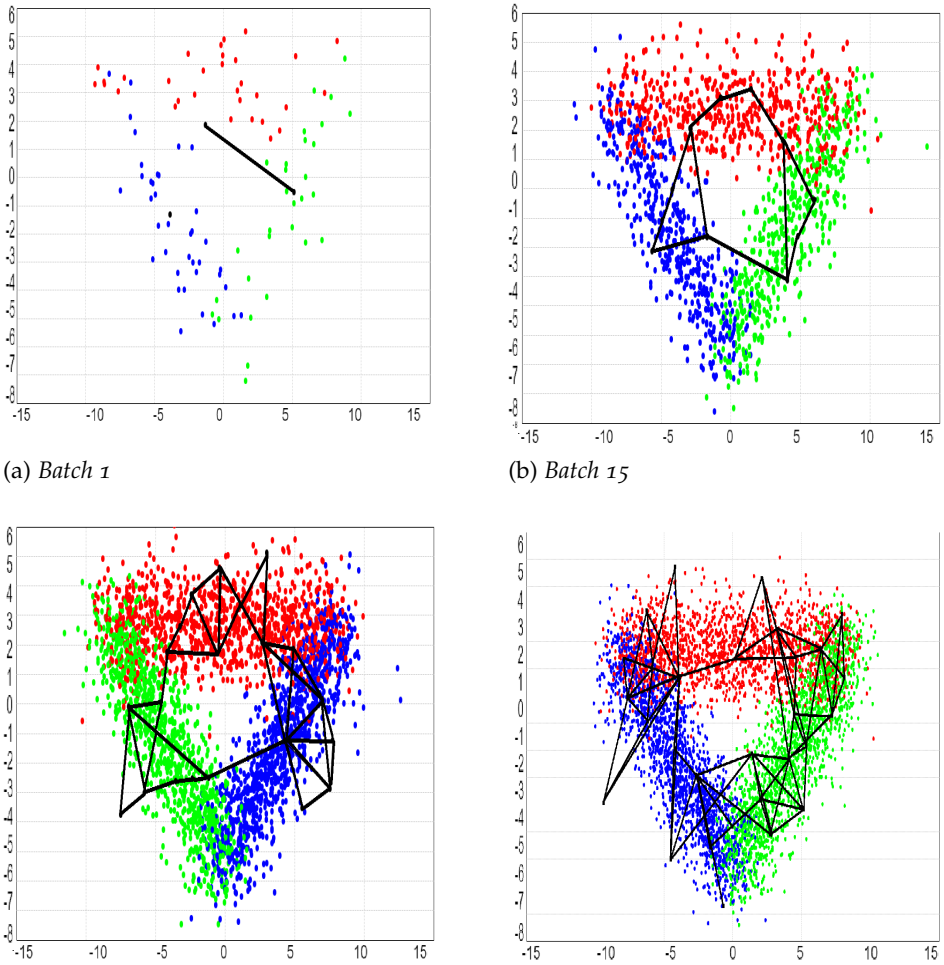


(a) *NMI* (b) *ARAND*

Figure 5.13 – *NMI and ARAND for all real datasets compared with results on reduced datasets.*

Clustering Evolution

Figure 5.14 shows an example of evolution of the graph S2G-Stream on *waveform* dataset (using Sammon's nonlinear mapping), as the data flows (colored points represent labelled data points and black points represent nodes of the graph with edges in black lines). We can clearly see that S2G-Stream, beginning with two randomly chosen nodes (Figure 5.14(a)), is able to recognize gradually the structure of the data stream (Figure 5.14(b,c)). At the end of the training we can observe that the topology recover all the data structure (Figure 5.14(d)). It is noticeable that our method manages to recognize the structures of the data stream and can separate these structures with the best visualization. It can also detect arbitrary-shaped clusters.



(a) *Batch 1* (b) *Batch 15*

(c) *Batch 30* (d) *Final Batch*

Figure 5.14 – *Evolution of graph creation of S2G-Stream on waveform dataset.*

Execution time

Figure (5.15) shows the execution times of S2G-Stream and the other stream clustering algorithms. We can notice that the CluStream algorithm has the shortest execution time, but our method is faster than DStream on all the datasets. In the meantime, S2G-Stream outperforms all the algorithms based on the results shown above. This result shows that S2G-Stream is nondominated across all datasets since no other algorithm yields faster computation times. In other words, no other algorithm can produce better results within equal or less time.

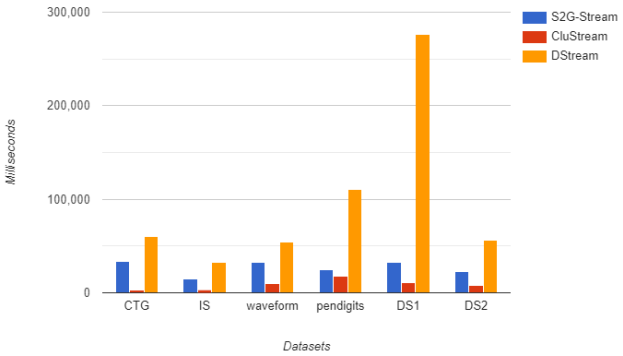


Figure 5.15 – *Execution time of S2G-Stream compared to other stream algorithms.*

Evolving Data Streams

In order to demonstrate the effectiveness of S2G-Stream in clustering evolving data streams, we evaluate in this subsection our method on the same datasets used above where the points are ordered by their class (i.e., data points of the first-class arrive first, then the ones of the second, third, etc.). The old classes disappear due to the use of the fading function. We report in Table (5.6) results of S2G-Stream on datasets used above with and without ordering classes. We ran both models in each case, on both ordered and unordered datasets, compared to CluStream and DStream based on NMI and ARAND measures. The results of this comparison are presented in Figure (5.16) and (5.17).

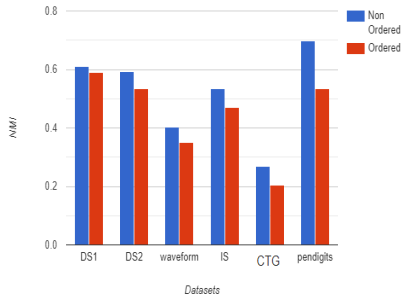
We can see that the values of NMI and ARAND for sorted datasets are slightly lowered then non-sorted dataset, but with no large differences. Unlike CluStream and DStream, where the performances of the clustering are greatly affected by order of the points. We conclude from this experiment that our method is well adapted for evolving data streams for most datasets.

Dataset	NMI		ARAND	
	Without Sorting	With Sorting	Without Sorting	With Sorting
Waveform	0.397±0.002	0.358 ± 0.11	0.137±0.007	0.135 ± 0.04
IS	0.550±0.05	0.419 ± 0,20	0.418±0.04	0,156 ± 0,010
CTG	0.270±0.009	0.205 ± 0,029	0.118±0.005	0,109 ± 0,06
pendigits	0.572±0,038	0.470 ± 0.08	0.408±0.060	0,170 ± 0,06
DS1	0.737±0.222	0.596 ± 0.16	0.299±0.106	0.178 ± 0.05
DS2	0.677±0.111	0.534 ± 0.08	0.303±0.071	0.214 ± 0,06

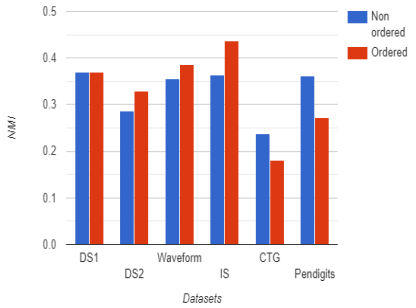
Table 5.6 – *Comparing results of S2G-Stream with and without sorted classes. The first value is the average of 10 repetitions followed by the standard deviation.*

Clustering over sliding windows

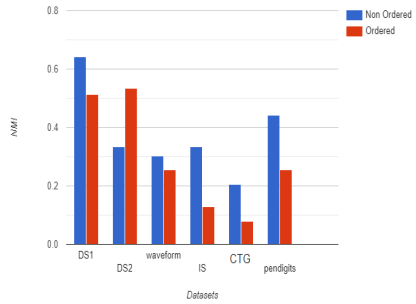
Most data stream algorithms consider the most recent data as more important and reflecting better the changes in the data distribution. Therefore, the notion of sliding window is introduced in order to analyze only the most recent data points and the model obtained from the previous ones.



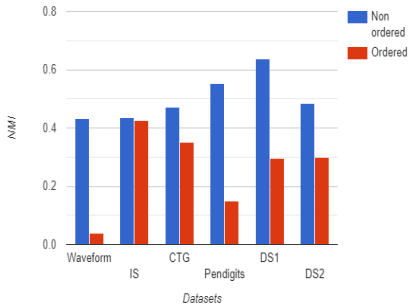
(a) *S2G-Stream(LWM)*



(b) *S2G-Stream(GWM)*



(c) *CluStream*



(d) *DStream*

Figure 5.16 – *NMI for different datasets with and without ordering classes compared with CluStream and DStream algorithms.*

Each window t contains P points, we consider that these P recent points contain some points from the previous window. Figure 5.18 illustrates the principle of the sliding window model.

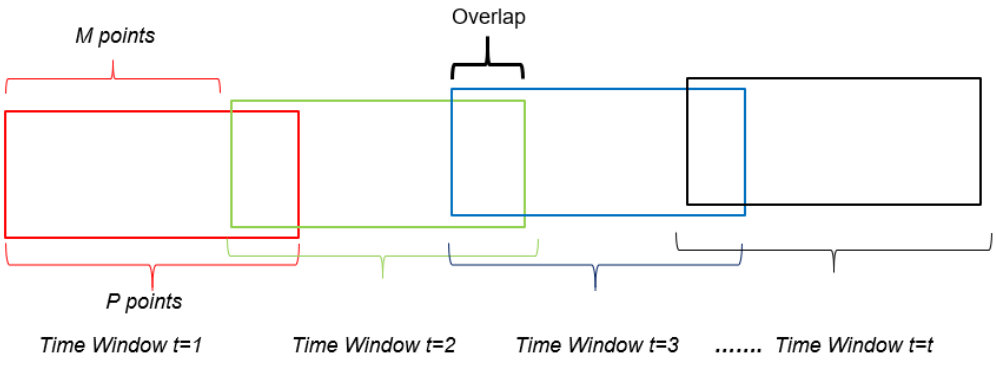
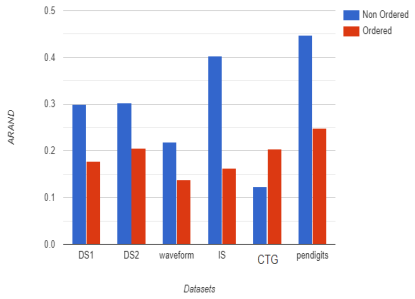
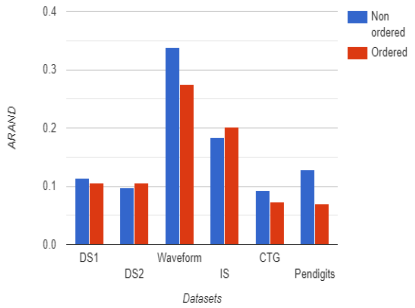


Figure 5.18 – *Sliding window model with windows overlap.*

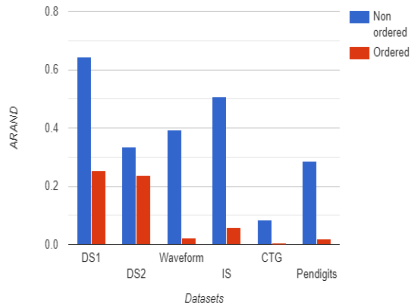
The overlap ratio between the two windows is the percentage of points that are kept from the previous window and re-used for learning with the new points. If at each step, the M oldest points are removed, and M points are appended, then the overlap ratio is defined as $(1 - M/P)$. We tested



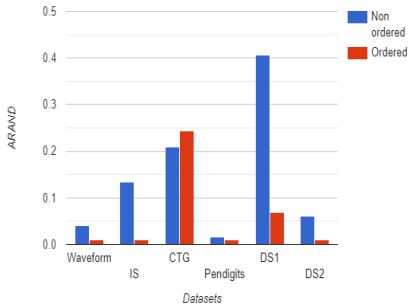
(a) *S2G-Stream(LWM)*



(b) *S2G-Stream(GWM)*



(c) *CluStream*



(d) *DStream*

Figure 5.17 – *ARAND for different datasets with and without ordering classes compared with CluStream and DStream algorithms.*

S2G-Stream(LWM) and S2G-Stream(GWM) with different overlap ratios for each dataset. The results are reported in Tables (5.7) and (5.8). We observe that the NMI and ARAND increase with the overlap ratio, since these datasets are small and medium-sized, the overlap between windows helps overcoming the problem of the small-sized dataset and as a result, enhances the performance of the method. While for large datasets, we can see a slight drop in performance since the overlap of the windows makes it difficult to learn from large datasets.

		Overlap ratio			
Datasets	Metrics	0%	25%	50%	75%
Waveform	NMI	0.397	0.397	0,400	0,407
	ARAND	0.137	0.214	0,217	0,216
IS	NMI	0.550	0.550	0.552	0.545
	ARAND	0.418	0.311	0.397	0.348
CTG	NMI	0.270	0.279	0.315	0.327
	ARAND	0.124	0.148	0.168	0.186
Pendigits	NMI	0.672	0.686	0.654	0.670
	ARAND	0.408	0.559	0.518	0.525
DS1	NMI	0.737	0.749	0.760	0.715
	ARAND	0.299	0.380	0.382	0.430
DS2	NMI	0.677	0.710	0.710	0.718
	ARAND	0.303	0.310	0.465	0.440

Table 5.7 – *NMI and ARAND of S2G-Stream(LWM) while changing the overlap percentage of sliding windows for each dataset.*

		Overlap ratio			
Datasets	Metrics	0%	25%	50%	75%
Waveform	NMI	0.355	0,283	0,283	0,308
	ARAND	0.399	0,281	0,281	0,314
IS	NMI	0.364	0,179	0,261	0,397
	ARAND	0.183	0,091	0,125	0,205
CTG	NMI	0.237	0,255	0,236	0,229
	ARAND	0.092	0,133	0,120	0,113
Pendigits	NMI	0.362	0,392	0,392	0,353
	ARAND	0.128	0,145	0,145	0,138
DS1	NMI	0.369	0,381	0,463	0,393
	ARAND	0.114	0,159	0,229	0,160
DS2	NMI	0.286	0,390	0,332	0,463
	ARAND	0.097	0,207	0,125	0,229

Table 5.8 – *NMI and ARAND of S2G-Stream(GWM) while changing the overlap percentage of sliding windows for each dataset.*

5.4 Conclusion

In this chapter, we have proposed S2G-Stream with two models of feature and block weighting (global and local), an efficient method for subspace clustering of an evolving data stream in an online manner. We used the weights obtained as scores to conduct more experiments. The subspace clustering method with the Global Weighting Model was used as a dimensionality reduction method. We proved the impact on the order of data point and also the overlapping of the windows to the clustering quality. Experimental evaluation demonstrates the effectiveness and efficiency of S2G-Stream in discovering clusters of arbitrary shapes and relevant features and blocks. The global model weighting gave us a comprehensive

view of the features and blocks that we used to select the best subset of features, and enhance the clustering performances.

The performance of S2G-Stream, in terms of clustering quality as compared to other relevant data stream algorithms are promising.

6 Improved Multi-objective Data Stream Clustering with Time and Memory Optimization

6.1 Introduction

AntTree [Azzag et al., 2003] is a hierarchical clustering method that models how ants form living structures and use this behavior to organize this data into a tree that is built in a distributed manner. Intuitively, each ant/data is located at the start of reliable support (tree root). The behavior of the ants then consists either in moving or in clinging to the structure to extend it and allow other ants to come and stick in their turn. This behavior is determined in particular by the similarity between the data and the local structure of the tree. The result is a tree-like organization of the data whose properties will allow us to determine a classification automatically and to have a visual overview of the tree.

The AntTree algorithm was proposed to deal with Data Stream, a kind of data that evolves and arrives in an unbounded stream. Analyzing data stream implies time and space constraints. The process of data stream clustering consists of creating compact and well-separated partitions from dynamic streaming data in only a single scan, using limited time and memory.

Most of the clustering techniques follow one objective function. However, every objective function represent a different property of the clusters, such as the compactness or the separateness of a cluster. When the algorithm assumes a homogeneous similarity measure over the entire data set, it becomes not robust to variations in cluster shape, size, dimensionality, and other characteristics [Handl and Knowles, 2007]. The Multi-Objective clustering methods (MOC) [Law et al., 2004] retrieve clusters by applying two or more objective functions. It uses a two-step process: 1) Generate multiple clustering solutions and store the optimal ones. 2) Construct an optimal partition based on the Pareto-set solutions. The following definitions are useful to understand MOC methods :

Definitions:

- *Dominated solutions:* a solution X is said to dominate a solution Y if $\forall j = 1, 2, ..., m, f_j(X) \leq f_j(Y)$, and there exists $k \in 1, 2, ..., m$ such that $f_k(X) < f_k(Y)$.
- *Pareto-optimal solutions:* a solution X is called Pareto-optimal if it is not dominated by any other feasible solutions. The set of non-dominated solutions is called Pareto-set.
- *Idle times:* in the case of a slow stream, time delays between data points can appear e.g., times where no data point is available. Traditional algorithms will stop and wait for new data points to process them. Figure 6.1 illustrates the concept of idle times.

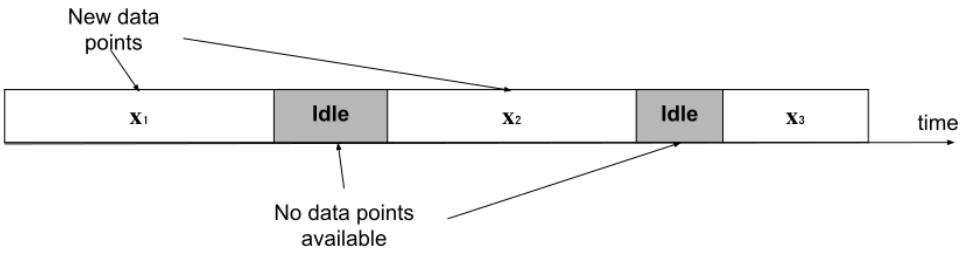


Figure 6.1 – *Idle times.*

6.2 AntTree Clustering

Ant-Tree algorithm [Azzag et al., 2003] produces a hierarchical structure in an incremental manner like how the ants join together. In this algorithm, each ant represents a single data point, and it moves in the structure according to the similarity $Sim(i, j)$ with the other ants already connected in the tree under construction. $Sim(i, j)$ is represented by the euclidean distance between two ants i and j. One should notice that this tree will not be strictly equivalent to a dendrogram as used in standard hierarchical clustering techniques: each node in our tree will correspond to one data while this is not the case in general for dendrograms, where data only correspond to leaves.

Starting from the support, materialized by a fictitious node f_0, the ants will progressively fix themselves on this initial point, then successively on the ants set at this initial point, and so on until all the ants are attached to the structure. During the construction of the structure, each ant f_i is either moving on the graph or connected to it. In the first case, f_i is free to move to a neighbor of the ant on which it is located (or to the support). In the second case, f_i will no longer be able to be released. Furthermore, we will consider the fact that each ant has only one outgoing link to other ants and cannot have more than L_{max} links connected to it from other ants (tree having at most L_{max} threads per node). Initially, all ants are placed on the f_0 support. They will each have a similarity threshold and a dissimilarity threshold, which are set to 1 and 0, respectively. An ant will connect under an existing node of the tree (ant f_{pos}) if it is sufficiently similar to this node but also dissimilar enough to the threads of the node: f_i will thus form a

subclass of f_{pos} which will be different from the other subclasses of f_{pos} (possibly already existing). Otherwise, f_i will move randomly in the tree, looking for another location to fix itself. As the f_i ant fails in its attempts to attach to the structure, it is made more tolerant in order to increase its chances of connecting to the next iteration concerning it: its similarity threshold is decreased, and its dissimilarity threshold is increased. The particular case of the f_0 support is treated as follows: an ant connects to the support if it is sufficiently dissimilar to other ants already connected directly to f_0. It means that a new class has just been built at the highest level of the tree. This class must be as distinct as possible from the other classes already created.
The algorithm ends when all the ants are connected. The sub-trees appearing at the first level of the tree, just below the support, will be interpreted as different classes. The properties of the tree can be analyzed visually and interactively (e.g., the classification error decreases as one goes down the tree). It is also possible to transform this tree into a dendrogram (by scrolling down the data placed on internal nodes to leaves.

6.3 Proposed Method

In this section, we introduce IMOC-Stream (Multi-Objective AntTree Clustering data stream). The algorithm is based on AntTree clustering and combines stream clustering and multi-objective clustering to create a Multi-objective stream clustering algorithm that satisfies two objective functions. It makes use of the hierarchical nature of AntTree and improve the clustering quality. We describe in the following sections the main properties of IMOC-Stream.

Clustering in a Streaming Context

We assume that the data stream consists in a sequence $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ of n (potentially infinite) elements, arriving at times $t_1, t_2, ..., t_n$, where $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$. Since the most recent data points are more important and reflect better the changes in the data distribution, we use temporal windows to consider only recent data for the clustering. A set of clustering solutions $\mathcal{S}$ is generated and updated for each window $\mathcal{S} = \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m$ where $\mathcal{C}_j$ is the j^{st} clustering solution and is represented by K clusters $\mathcal{C}_j = c_1, c_2, .., c_K$. Each cluster c is represented by a prototype $\mathbf{w}_c$ where $\mathbf{w}_c = (w_c^1, w_c^2, \ldots, w_c^d)$ and d is the dimension of the data. Each cluster is associated with a weight π_c that decreases over time based following a fading function.

When the first batch of data arrives in the first time window, we create the tree as a clustering solution according to Section 6.2, and this solution is stored in the Pareto-set. From the same batch of data we initialize several solutions using K-means [Pelleg et al., 2000] with different K, GNG [Fritzke, 1995], DBScan [Ester et al., 1996]. The parameter settings of these algorithms are reported in Table 6.1. The generated solutions are combined with the tree solution by the mutation and crossover operators, and the results are added to the solutions-set. We compute the objective func-

tion values for each solution-set and store the non-dominated solutions in the Pareto-set.

Algorithm	Parameters	Source
GNG	$epochs = 30$	Smile Package[1]
DBSCAN	$minPts = 20$ $radius = 10$	Smile Package[2]
K-means	K vary from 2 to 15	Clustering4Ever[3]
Ant-tree	$L_{max} = 10$	Clustering4Ever[4]

Table 6.1 – *Parameter settings for the used algorithm*

After the Initialization phase and for each new window of data points, each point in the current window is assigned to the closest center c_{ij} in each clustering solution C_j in the pareto-set. The distance calculated between the data points and the centers is the euclidean distance. We note that for each clustering solution, a point can be assigned to only one cluster. After all points being assigned, we update the clustering solutions with the new assigned points. We compute the objective functions values f_i and we update the pareto-set. If the system idles, the method combines the solutions in the pareto-set using the genetic operators and calculates the objective values of the new generated solutions. At the end of each iteration, the pareto-set contains a set of non-dominated solutions. At the end of the process, a set of non-dominated clustering solutions is stored. These solutions are equally good mathematically. We used an internal quality measures Davies Bouldin [Davies and Bouldin, 1979] to select the best solution among the Pareto-set.

AntTree with Tree Aggregation

To deal with the memory constraints encountered when analyzing data streams, we propose a new representation of the tree to prevent storing all the data points and to reduce the memory allocation. The tree is initialized from the data points in the first window following the AntTree algorithm described in section 6.2. After placing all the points, we compute the prototypes w of each cluster as the average of the points assigned to this cluster. All the points are discarded, and only the tree with the prototypes is stored in the memory. For the next windows, we assign the new data points to each cluster and update the prototype $\mathbf{w}_c$ as follow:

$$\mathbf{w}_c^{(t+1)} = \frac{\mathbf{w}_c^{(t)} n_c^{(t)} \gamma + \mathbf{z}_c^{(t)} m_c^{(t)}}{n_c^{(t)} \gamma + m_c^{(t)}} \tag{6.1}$$

Where $\mathbf{w}_c^{(t)}$ is the previous prototype, $n_c^{(t)}$ is the number of points assigned to the cluster, $\mathbf{z}_c^{(t)}$ is the new prototype computed only from the current window. $m_c^{(t)}$ is the number of points assigned to the cluster c in the current window: $n_c^{(t+1)} = n_c^{(t)} + m_c^{(t)}$. γ is the decay factor that decreases over time to give more importance to most recent data $0 < \gamma < 1$. If $\gamma = 1$ all data will be used from the beginning; $\gamma = 0$ only the most recent data

will be used.

If a point is not assigned to a cluster, it becomes a prototype of a newly created cluster. Figure 6.2 illustrates tree representation and aggregation.

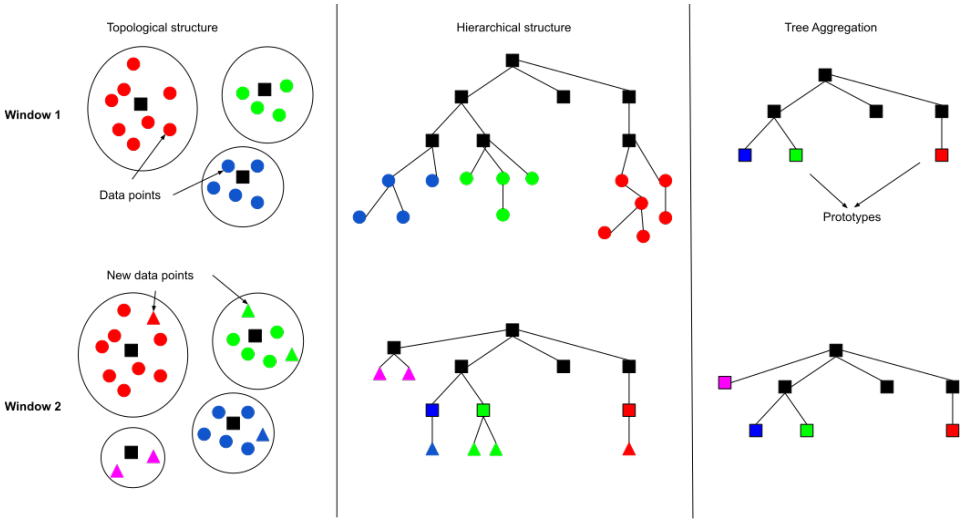


Figure 6.2 – *Topological and Hierarchical representation and tree aggregation process. The circles represent the data points, the squares represent the prototypes ans the triangles represent the new data points from the current window.*

Fading Function

Most data stream algorithms consider the most recent data as more important and reflect better the changes in the data distribution. For that, we consider a *Fading* function, in which the weight of each cluster decreases exponentially with time t by introducing a decay factor parameter $0 < \gamma < 1$.

$$\pi_c^{(t+1)} = \sum_{i=1}^{n_c} 2^{\gamma(t-t_0)}, \tag{6.2}$$

where n_c is the number of points assigned to the cluster c at the current time t. If the weight of a node is below a threshold value, this cluster is considered outdated and removed.

Evolutionary Representation and Functions

Most of the Multi-Objective clustering methods use an evolutionary representation for the clustering solutions as their use of population enables the variation of solutions and makes it easier to keep a population of clustering solutions and apply genetic operators. However, the use of such representation requires the following concepts:

- Choosing an evolutionary encoding to represent a clustering solution.
- The generation of the initial population by an effective initialization scheme.
- Suitable genetic operators to variate the solutions.

- Choosing two or more objective functions as a fitness function to choose the non-dominated solutions.
- Developing a technique to obtaining a single clustering solution for the Pareto-set (leader selection method).

The choice of these components is crucial for the clustering quality and the algorithm scalability. In the next sections, we describe the components we chose after extensive experiments to deal with the requirements presented above.

Genetic Representation

Many representations were presented in the previous MOC methods [Mukhopadhyay et al., 2015]. However, these representations are not suitable for data stream clustering since data points can not be stored and have to be processed in one pass. Therefore, we chose a new genetic representation that facilitates the clustering update as the data flows. Each clustering solution is represented by a chromosome, which is an array of $K \times d + 2$, where d is the dimension of the data. The first and second components are the objective values for this solution. The last K components represent the clusters. Each cluster is represented by a prototype $\mathbf{w}$ of d elements. Figure 6.3 illustrates the clustering representation and conversion.

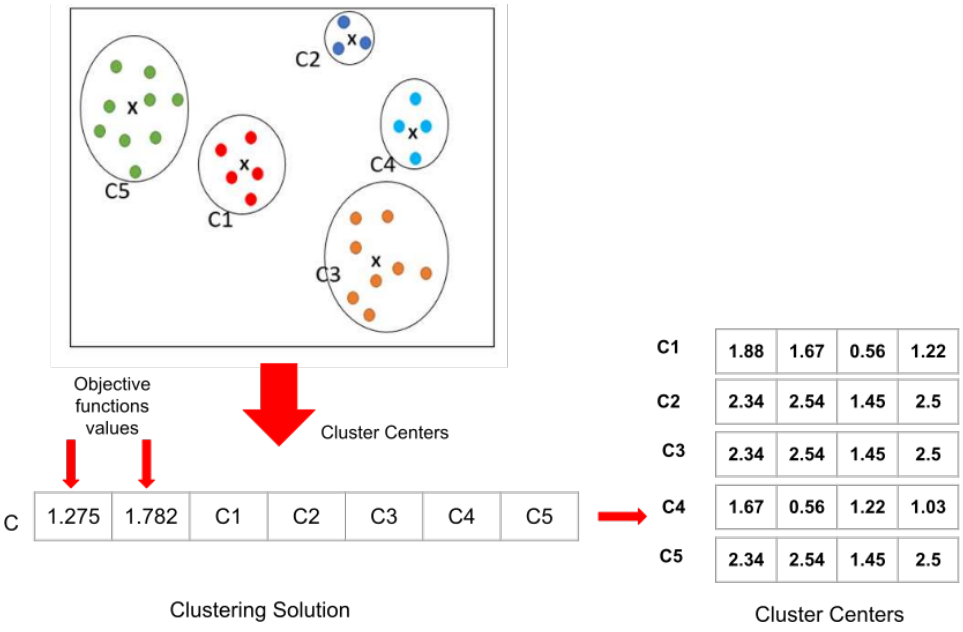


Figure 6.3 – *Clustering solution representation and conversion.*

Population Initialization

In each time window of the data stream, a set of clustering solutions is created and stored. Our algorithm does not require these solutions to have the same number of clusters. A first population is created from the first window using the AntTree algorithm combined with other solutions generated by several algorithms (K-means [Pelleg et al., 2000] with different K, GNG [Fritzke, 1995], DBScan [Ester et al., 1996]). Those algorithms were

chosen after extensive experimentation due to their ability to do a local search. The solutions given are encoded following the scheme described in Figure 6.3. We select the best solutions from this population to create new clustering solutions following the genetic operators Crossover and Mutation described in section 6.3. After the first population initialized, we compute objective functions for each clustering solution and store the Pareto-optimal solutions into the Pareto-set. For each window of the data stream, the new data points belonging to the current window are used to update the solutions in the Pareto-set and to create new solutions. The Pareto-set is then updated with the non-dominated solutions. We describe the initialization and update scheme in Figure 6.4.

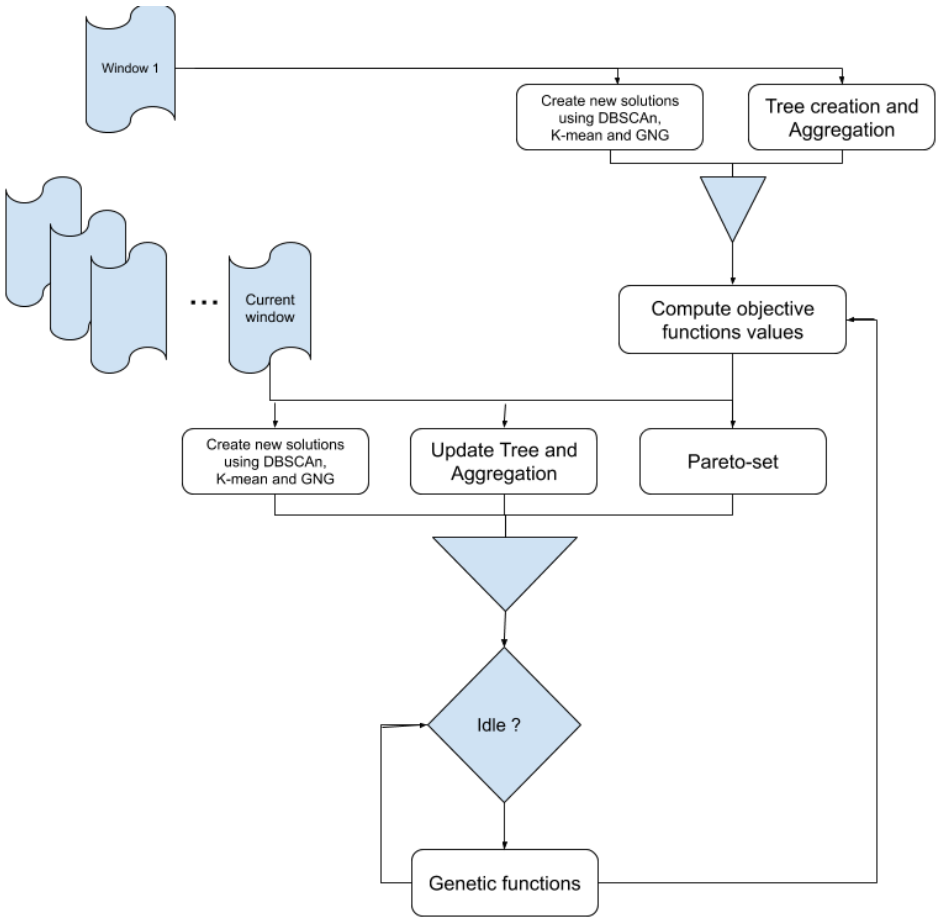


Figure 6.4 – *Initialization and update scheme.*

Genetic Functions

Genetic operators are essential for MOC methods as they enable the variety and diversity of the clustering solutions. For our method, we use two genetic operators: Crossover and Mutation, to explore more solutions. The use of those operators helps find a better solution by combining the optimal solutions obtained from the other algorithms.

- *Crossover*: We used the single point crossover [Whitley, 1994] in this chapter due to its Independence of the ordering of genes. The goal of the crossover operator is to create new clustering solutions from

the two-parent solutions. First, we randomly select the Pareto-set two solutions that have respectively K_1 and K_2 clusters. We choose randomly a crossover point i, as the number of clusters may vary, i must satisfy $1 < i < min(K_1, K_2)$.

The first resulted clustering solution from the crossover is composed of cluster centers from 1 to i of the solutions with $min(K_1, K_2)$ cluster centers, and i+1 to K of the second clustering solution. The second resulted clustering solution is composed of the cluster centers $i+1$ to $min(K_1, K_2)$ from the first solution and of cluster centers 1 to i from the second solution. Figure 6.5 explains the process of crossover of two clustering solutions.

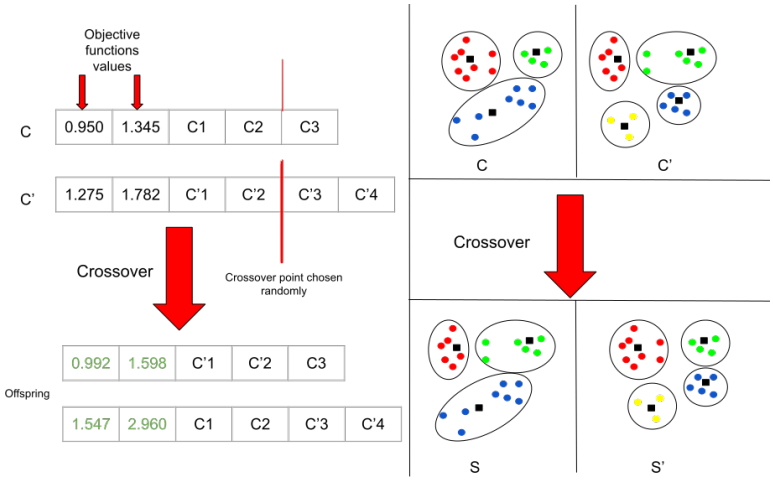


Figure 6.5 – *Crossover of two clustering solutions. The figure on the left represents the prototypes and the one on the right represents the topological clusters, the squares represent the prototypes and the circles are the data points. The data points are added to illustrate, in the clustering process, no data point is kept in the memory.*

- *Mutation*: We use the random resetting mutation operator [Mitchell, 1998] to change randomly some values of a cluster center in a clustering solution to explore global solutions. We select a clustering solution C from the Pareto-set, then from each cluster center in C, we randomly select μ position values. For a value v, a number $0 < \varrho < 1$ is generated and the value v is updated as follows:

$$v \pm \varrho * v,$$

The '+' or '-' signs occur with equal probability. Figure 6.6 illustrates the process of mutation of a clustering solution.

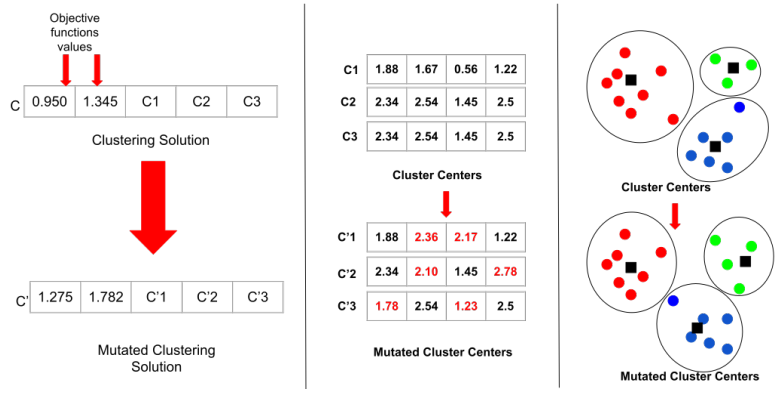


Figure 6.6 – *Mutation of a clustering solution.* $\mu = 50\%$

Both operators are applied during idle times on the solutions from the Pareto-set. The solutions are selected based on their fitness score, equal to $(-separateness + compactness)$. We select σ clustering solutions and apply the genetic operators.

Objective Functions

One of the important aspects of MOC is the choice of suitable objective functions that are to be optimized simultaneously. For each clustering solution, several quality measures exist. The the goal is to have distinct clusters (separateness) that are the most dense in terms of the data points they contain (compactness). To satisfy these requirements, we introduce two objective functions *compactness* and *separateness*. The combination of both objective functions allows us to have arbitrary shaped clusters.

- *Compactness*: the compactness of a clustering solution reflects the overall intra-cluster size of the data and has to be minimized. The compactness of a clustering solution in a streaming context is computed as follows:

$$Compactness_{\mathcal{C}}^{t+1} = \gamma Compactness_{\mathcal{C}}^{t} + \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} \delta(\mathbf{x}_i, \mathbf{w}_{\phi(\mathbf{x}_i)}) \tag{6.3}$$

  Where $\mathcal{X}^{(t+1)}$ is the current window and $\phi(\mathbf{x}_i)$ is the index of the cluster where $\mathbf{x}_i$ belongs. $\delta(\mathbf{x}, \mathbf{w}_{\phi(\mathbf{x}_i)}$ is the euclidean distance between the data point $\mathbf{x}$ and $\mathbf{w}_{\phi(\mathbf{x}_i)}$. γ is the decay factor that decreases over time to give more importance to most recent data. The points of the previous windows are not kept, $Compactness_{\mathcal{C}}^{t}$ has been computed in the previous window with the previous prototype $\mathbf{w}$.

- *Separateness*: the separateness of a clustering solution is the mean distance between clusters. It reflects the inter-cluster similarity and should be maximized. The separateness of a cluster is the shortest distance between a data point in this cluster and another data point of his neighborhood belonging to another cluster. In a streaming context, the separateness is computed as follows:

$$Separateness_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} (min_{\mathbf{x}_i \in c, \mathbf{x}_j \in \mathcal{K}_i, \mathbf{x}_j \notin c} \delta(\mathbf{x}_i, \mathbf{x}_j)) \tag{6.4}$$

Where $\mathcal{K}_i$ is the neighborhood of the data point $\mathbf{x}_i$ belonging to the cluster c. The neighborhood of a node is determined through the AntTree method. The neighborhood of a cluster is the directly connected nodes to this one on the tree.

Solution Selection

At the end of the online phase, a set of non-dominated solutions is stored in the Pareto-set. These non-dominated solutions are equally good mathematically. We used an internal quality measures Davies Bouldin [Davies and Bouldin, 1979] to select the best solution among the Pareto-set. The choice of an internal index is because the data might not be labeled. We sort all the solutions by their fitness (internal measures values), and we choose the best one as an output of the algorithm. Davies Bouldin index helps identify sets of clusters that are compact and well separated. The Davies-Bouldin index is described in Section 3.2.1.

$$DBI = \frac{1}{K}\sum_{i=1}^{K} max_{i,j=1,..,K;j\neq i}\frac{d(x_i, C_i) + d(x_j, C_j)}{d(C_i, C_j)} \tag{6.5}$$

$d(x_i, C_i)$ is the distance between the data point x_i, and its cluster C_i and K is the number of clusters. DBI varies between 0 (best clustering) and $+\infty$ (worst clustering).

Improved MOC-Stream Algorithm

IMOC-Stream is an extension of Multi-objective clustering for data stream to optimize computation time and memory allocation. It starts with creating a first clustering solution using the AntTree algorithm. On the contrary of the original algorithm where all the data points are stored, we introduced a new tree aggregation method to store only a synopsis of the data. The clustering solution is encoded and combined with different solutions obtained by different algorithms to create a population of solutions. The objective function values are computed for each solution, and only the non-dominated solutions are added to the Pareto-set. Then, we apply crossover and mutation on the best solutions selected from the Pareto-set and add the obtained solutions to the population. For each time window, the next point from the stream is mapped into the tree, the prototypes are computed, and only the aggregated tree is stored. We update the weights of the nodes and remove the outdated ones. If the stream idles, we apply genetic operators to generate more solutions. At the end of each time window, we compute the objective function values, select the non-dominated solutions, and update the Pareto-set. In the offline phase, we compute the internal index Davies Bouldin of each potential solution and select the optimal one as an output for this algorithm. In summary, the algorithm of IMOC-Stream presented in this chapter is described in Algorithm 11.

Algorithm 11 Improved MOC-Stream Algorithm

Result : Optimal clustering solution

From the first window: initialize the tree using AntTree algorithm and perform tree aggregation following Figure 6.2;
Generate several clustering solutions using K-means with different K, GNG, and DBScan with different parameters;
Encoding the clustering solutions following scheme in Figure 6.3;
Apply Crossover and Mutation following Figures 6.5 and 6.6 respectively. Add new solutions to the population of chromosomes;
Compute objective functions following equations (6.3) and (6.4). Store non-dominated solutions in the Pareto-set;

while *There is data available* **do**
    Map each point into the tree and compute prototypes following Equation(6.1);
    For each clustering solution in the pareto-set, assign each point to the closest cluster;
    Update each cluster in each clustering solution using the new points assigned as described in Equation (6.1);
    Update weights of nodes following Equation (6.2) and remove the outdated nodes;
    Compute objective functions of the clustering solutions in the pareto-set and the new solutions generated. Update the pareto-set with the new non-dominated solutions ;
    while *Idle* **do**
        Select best clustering solutions from Pareto-set based on their objective values ;
        Apply Crossover and Mutation following Figures 6.5 and 6.6 respectively. Add new solutions to the population of clustering solutions;
    end
end
Select best solution among the pareto-set solutions as described in section 6.3

6.4 Experiment Results

Datasets and Quality Criteria

The IMOC-Stream method described in this article was implemented in Scala programming language and will be available on Clustering4Ever GitHub repository[5]. We evaluated the clustering quality of IMOC-Stream on several real [Frank and Asuncion, 2010] and synthetic[6] datasets. We describe the datasets in Table 6.2. The mutation rate μ is set to 20% and the number of selected clustering solutions to the crossover and mutation is set to 10.

For the quality measures, we used the internal measures (NMI) [Strehl

[5]https://github.com/Clustering4Ever/Clustering4Ever
[6]https://www.sites.google.com/site/nonstationaryarchive/

Dataset	Instances	Features	Classes	$\|Window\|$
powersupply	29,928	2	24	100
HyperPlan	100,0000	10	5	1000
Covertype	581,102	10	23	1000
Sensor	2,219,802	4	54	10000
1CDT	16000	2	2	100
1CSurr	55280	2	2	1000
4CR	144000	2	1	1000
GEARS-2C-2D	200000	2	2	10000

Table 6.2 – *Description of datasets used in experimentation*

and Ghosh, 2002] and the Adjusted Rand index (ARAND) [Hubert and Arabie, 1985] described in Section 3.2.2.

Experimental Settings

Assuming large high-dimensional data arrives as a continuous stream, IMOC-Stream divides the streaming data into batches and processes each batch continuously. The batch size depends on the available memory and the size of the original dataset the size of the window for each dataset is shown in Table 6.2. We set the time interval between two batches to 1 second and the parameter γ to 0.7.
To show the effectiveness of our method, we compared it to five well known stream algorithms: *StreamKM++* [de Andrade Silva and Hruschka, 2011], *DStream* [Tu and Chen, 2009], *DBStream* [Hahsler and Bolaños, 2016], *DenStream* [Cao et al., 2006] and *CluStream* [Aggarwal et al., 2003] from R package streamMOA[7]. We repeated our experiments with different initialization and have chosen those giving the best results. Table 6.3 shows the optimal parameter configurations.

Algorithms	Parameters	Initialization
DStream	$gridsize$	0.9
	λ	0.001
	$gaptime$	1000
	Cm	3
DBStream	r	1.8
	λ	0.001
	$gaptime$	1000
	Cm	2.5
DenStream	ϵ	0.4
	μ	1.605
	β	0.275
CluStream	t	2

Table 6.3 – *Optimal parameter configurations for the algorithms used for the experimentation. For IMOC-Stream, the decay factor is fixed to 0.7.*

[7]https://github.com/mhahsler/streamMOA

Clustering Evaluation

The results of IMOC-Stream on the datasets described above compared to the different algorithms are reported in Tables 6.4 and 6.5. The value value of NMI and ARAND is the average value of ten runs. It is noticeable that IMOC-Stream gives better results than all the other methods. These results are due to the fact that our method optimizes two objective functions to maximize intra-cluster similarity and minimize inter-cluster similarity at the same time, which gives us a compact and well-separated clusters. The use of different algorithms to create a population of solutions allow IMOC-Stream to explore better solutions and escape the local minima. Another critical point is the use of the genetic parameters to combine the best solutions and explore the potential local solutions. The other algorithms are sensitive to the initialization of the settings, which justify why our algorithm yields better results since it has no input parameters. Finally, we noticed that the DStream algorithm gives better results compared to the other algorithms used in this experimentation since it is adapted to large datasets.

For synthetic datasets, IMOC-Stream also gave better results than the different stream algorithms except for StreamKM++ on the 1*CSurr* dataset. These results are due to the optimal choice of the K for StreamKM++, which makes it find the exact number of clusters with synthetic datasets and gives better results. The number of clusters is not predefined in IMOC-Stream, but it still manages to find approximately the right amount of clusters.

Dataset	Metrics	IMOC-Stream	StreamKM++	DStream	DBStream	DenStream	CluStream
powersupply	NMI	**0.466 ± 0.03**	0,232 ± 0,05	0,403 ± 0,06	0,056 ± 0,01	0.055 ± 0.01	0.196 ± 0.05
	ARAND	**0.144 ± 0.03**	0,034 ± 0,01	0,049 ± 0,01	0.001 ± 0.00	0.002 ± 0.00	0.032 ± 0.01
Sensor	NMI	**0.723 ± 0.00**	0.151 ± 0.03	0.274 ± 0.07	0.060 ± 0.01	0.032 ± 0.00	0.024 ± 0.00
	ARAND	**0.192 ± 0.00**	0.074 ± 0.02	0.034 ± 0.01	0.006 ± 0.00	0.032 ± 0.00	0.006 ± 0.00
Covertype	NMI	**0.509 ± 0.03**	0.113 ± 0.03	0.310 ± 0.06	0.048 ± 0.001	0.482 ± 0.12	0.295 ± 0.07
	ARAND	**0.433 ± 0.10**	0.165 ± 0.02	0.254 ± 0.08	0.002 ± 0.003	0.198 ± 0.06	0.339 ± 0.11
HyperPlan	NMI	**0.168 ± 0.01**	0.026 ± 0.00	0.140 ± 0.03	0.002 ± 0.00	0.026 ± 0.00	0.014 ± 0.01
	ARAND	**0.041 ± 0.00**	0.035 ± 0.00	0.093 ± 0.02	0.001 ± 0.00	0.027 ± 0.00	0.019 ± 0.00

Table 6.4 – *Comparing IMOC-Stream with different algorithms on real datasets. The first value is the average of 10 repetitions and the value after ± is the standard deviation.*

Dataset	Metrics	IMOC-Stream	StreamKM++	DStream	DBStream	DenStream	CluStream
1CDT	NMI	**0.990 ± 0.07**	0.759 ± 0.03	0.691 ± 0.10	0.631 ± 0.28	0.208 ± 0.05	0.621±0.06
	ARAND	**0.970 ± 0.09**	0.679 ± 0.02	0.667 ± 0.14	0.610 ± 0.30	0.086 ± 0.05	0.583±0.09
1CSURR	NMI	0.481 ± 0.00	**0.534 ± 0.12**	0.136±0.17	0.031±0.02	0.150±0.05	0.409±0.1
	ARAND	0.248 ± 0.01	**0.529 ± 0.17**	0.041±0.19	0.02±0.07	0.017±0.07	0.384±0.11
4CR	NMI	**0.957 ± 0.00**	0.705±0.01	0.804 ± 0.02	0.868 ± 0.03	0.183 ± 0.03	0.502 ± 0.03
	ARAND	**0.954 ±0.00**	0.497 ± 0.02	0.793 ± 0.03	0.881 ± 0.02	0.006 ± 0.00	0.408 ± 0.02
GEARS_2C_2D	NMI	**0.654 ± 0.02**	0.543±0.03	0.160±0.12	0.001±0.00	0.021±0.02	0.301±0.02
	ARAND	**0.643 ± 0.01**	0.449±0.03	0.154±0.17	0.0001±0.00	0.010±0.01	0.219±0.02

Table 6.5 – *Comparing IMOC-Stream with different algorithms on synthetic datasets. The first value is the average of 10 repetitions and the value after ± is the standard deviation.*

Clustering High Dimensional Data

The curse of dimensionality refers to various phenomena that arise when clustering data in high-dimensional spaces. Most of the clustering algo-

rithms suffer from the curse of dimensionality. This is due to many factors like the high number of parameters to set or the algorithm's high complexity. To prove our method's effectiveness on clustering high dimensional datasets (HDD), we tested it on 6 HDD's from [Fränti et al., 2006]. The dimensions (number of features) of these datasets vary from 32 to 1024 while the number of instances is 1024 and the number of classes equal to 16. We compared our results with different stream algorithms based on NMI and ARAND measures. The results are reported in Table 6.6. The results show that our method outperforms all the other methods in terms of NMI and ARAND. These results are because our algorithm does not require parameter settings and uses linear genetic functions to enhance the quality, unlike the other algorithms. The pre-defined parameter and the use of costly processes and algorithms (like DBSCAN for DBStream and DenStream) make the algorithms slower and not robust when dealing with HDD. The genetic operators and the update of the solutions in our method are performed linearly, making these functions not costly in the computation time.

Dataset	Metrics	IMOC-Stream	StreamKM++	DStream	DBStream	DenStream	CluStream
dim032	NMI	**0.600 ± 0.01**	0.500 ± 0.04	0.051 ± 0.03	0.421 ± 0.06	0.062 ± 0.02	0.211 ± 0.02
	ARAND	**0.227 ± 0.01**	0.199 ± 0.04	0.021 ± 0.01	0.139 ± 0.02	0.003 ± 0.00	0.0215 ± 0.01
dim064	NMI	**0.675 ± 0.01**	0.546 ± 0.00	0.037 ± 0.01	0.522 ± 0.02	0.104 ± 0.04	0.184 ± 0.02
	ARAND	**0.380 ± 0.00**	0.252 ± 0.01	0.005 ± 0.04	0.339 ± 0.01	0.017 ± 0.01	0.012 ± 0.01
dim128	NMI	**0.691 ± 0.01**	0.571 ± 0.04	0.136 ± 0.01	0.531 ± 0.02	0.147 ± 0.05	0.191 ± 0.01
	ARAND	**0.418 ± 0.02**	0.386 ± 0.12	0.056 ± 0.02	0.321 ± 0.01	0.090 ± 0.03	0.003 ± 0.01
dim256	NMI	**0.777 ± 0.01**	0.575 ± 0.07	0.078 ± 0.01	0.391 ± 0.02	0.147 ± 0.03	0.171 ± 0.00
	ARAND	**0.487 ± 0.00**	0.377 ± 0.16	0.055 ± 0.03	0.265 ± 0.01	0.045 ± 0.01	0.004 ± 0.02
dim512	NMI	**0.788 ± 0.01**	0.606 ± 0.04	0.115 ± 0.01	0.329 ± 0.10	0.112 ± 0.12	0.145 ± 0.01
	ARAND	**0.540 ± 0.00**	0.538 ± 0.03	0.073 ± 0.02	0.478 ± 0.12	0.045 ± 0.09	0.002 ± 0.02
dim1024	NMI	0.855 ± 0.00	0.774 ± 0.02	0.112 ± 0.03	0.305 ± 0.09	0.110 ± 0.05	0.150 ± 0.00
	ARAND	0.717 ± 0.01	0.634 ± 0.02	0.020 ± 0.03	0.414 ± 0.08	0.041 ± 0.07	0.005 ± 0.01

Table 6.6 – *Comparing IMOC-Stream with different algorithms on HDD datasets. The first value is the average of 10 repetitions and the value after ± is the standard deviation.*

Clustering Histopathological Images

To prove the efficiency of our method when dealing with high dimensional datasets, we tested our method on hispathological images. High-resolution histopathology images provide reliable information differentiating abnormal tissues from normal ones, and thus, it is a vital technology for recognizing and analyzing cancers. However, in histopathology cancer image analysis, if a small part of the image is considered as cancer tissues, pathologists should diagnose the histopathology as positive.

In this chapter, we consider the clustering of patches of images. We extract 224 x 224 patches from each image using a sliding window with an overlapping rate r =0.7. Our method is then applied to the pieces of tissue to create similar clusters. The purpose of this treatment is to extract similar patches and to group them according to their similar morphology. Those groups of patches can be classified into cancer or non-cancer instead of categorizing the whole set of patterns. Figure 6.7 illustrates the patches' extraction from a tissue image.

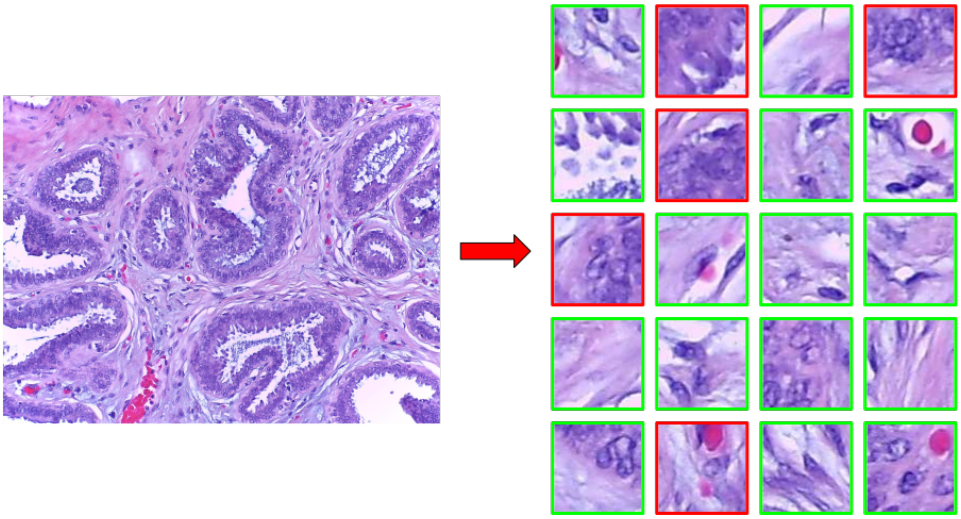


Figure 6.7 – *Patches extraction from a tissue image. The red rectangles represent cancer tissues, and the green ones represent non-cancer tissues.*

Breast cancer is one of the leading cancer-related death causes worldwide, especially on women. However, early diagnosis significantly increases treatment success. For the purpose of early diagnosis, proper analysis of histology images is essential. Individually, during the diagnosis procedure, specialists evaluate both overall and local tissue organization via whole-slide and microscopy images.

The ICIAR 2018 challenge has proposed two types of datasets: a. the first is composed of microscopic images and b. the second contains whole slide images. In this research, we used the ICIAR 2018-A2 dataset, which includes 400 breast histology images ($2048px \times 1536px$) that have been digitized under 200 $\times$ magnification. Two pathologists have annotated these images into four classes: normal, benign, in situ carcinoma, and invasive carcinoma. We extracted 224×224 patches from each image using a sliding window with an overlapping rate $r = 0.7$. The number of extracted patches is 43210.

Figure 6.8 represents an example of clustering by IMOC. Besides the similarity between the patches in the same cluster, we notice that in some groups, most of the patches have anomalies. This means that this group can be categorized as cancer tissues. The goal of this experiment is to provide groups of similar tissue to help researchers detect cancer-based on a group of patches instead of analyzing the whole set of patches, which can minimize the computational time.

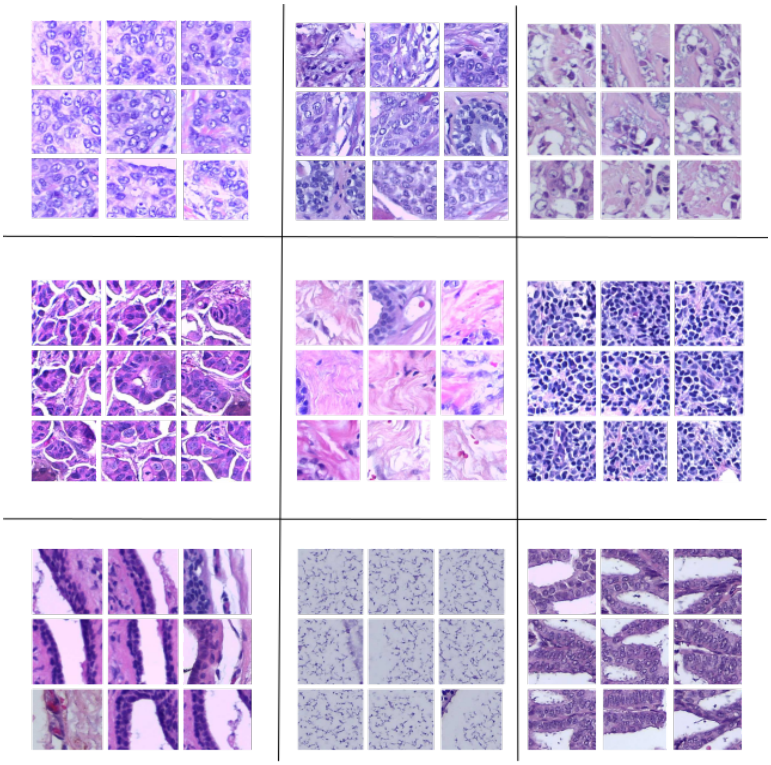


Figure 6.8 – *Example of clusters obtained by IMOC. Each group of 9 patches represents a cluster.*

Clustering Evolution

Figure 6.9 shows an example of the evolution of IMOC-Stream clustering on *1CDT*, *4CE-V1*, *1CH* and *2CDT* datasets. Each line represents an evolution of clustering for a particular dataset. These figures are generated during the clustering process. We picked three partitionings at random iterations for each dataset. For each time window, the distribution of the incoming data points changes. With its Multi-Objective capability and the fading function's use, IMOC-Stream manages to recognize the structures of the data stream and can separate these structures with the best visualization. It can also detect arbitrary shaped, compact, and well-separated clusters. We note that the number of clusters does not necessarily stay the same, but the best K is automatically chosen.

Arbitrary Shaped Clusters

Figure 6.10 represents the cluster detection for the t4.9k, Compound, and Path-based datasets[8]. We can see from this figure that our method manages to find clusters of arbitrary shapes and provide a good separation of the clusters. The other streaming clustering methods are unable to find clusters of arbitrary shapes (only spherical clusters may be found). The IMOC-Stream method is also able to find noise points due to the use of a density clustering method (DBSCAN).

[8]http://cs.joensuu.fi/sipu/datasets/

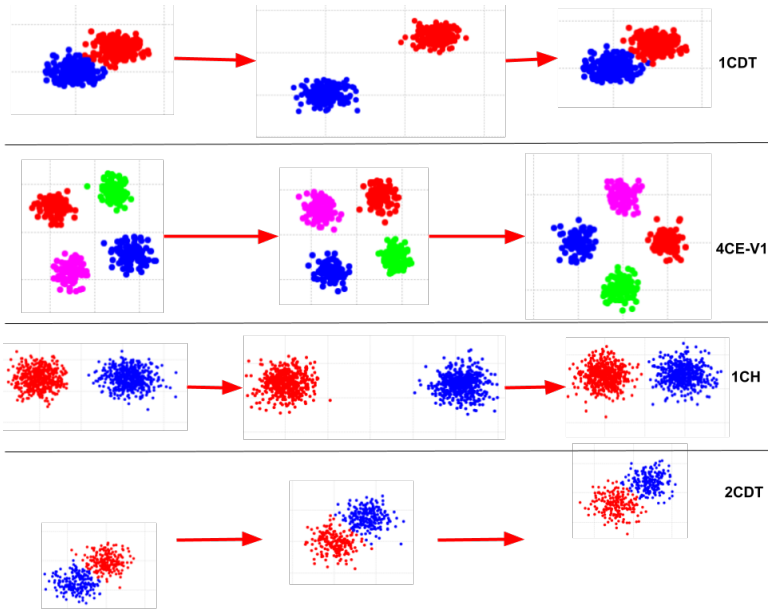


Figure 6.9 – *Clustering evolution of 1CDT, 4CE-V1, 1CH, 2CDT datasets. Each color represents a cluster. Each line represents the evolution of a clustering with one dataset.*

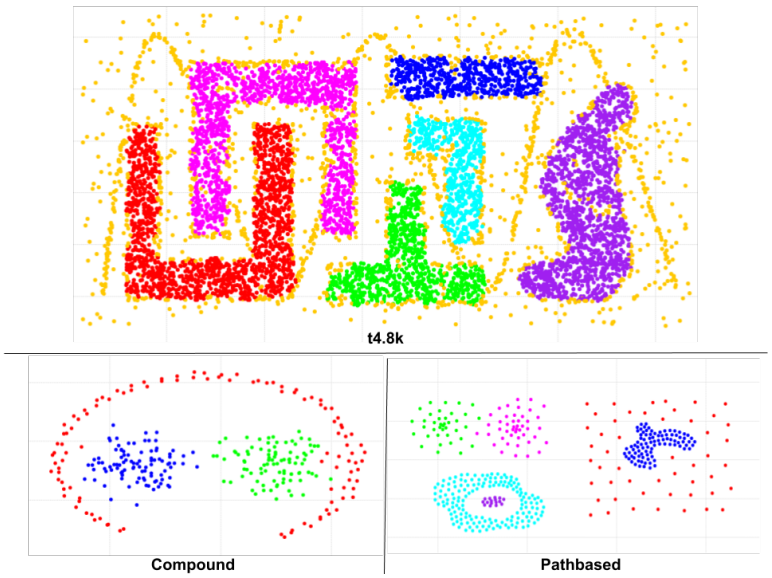


Figure 6.10 – *Examples of detection of arbitrary shaped clusters by IMOC algorithm.*

Time and Memory Complexity

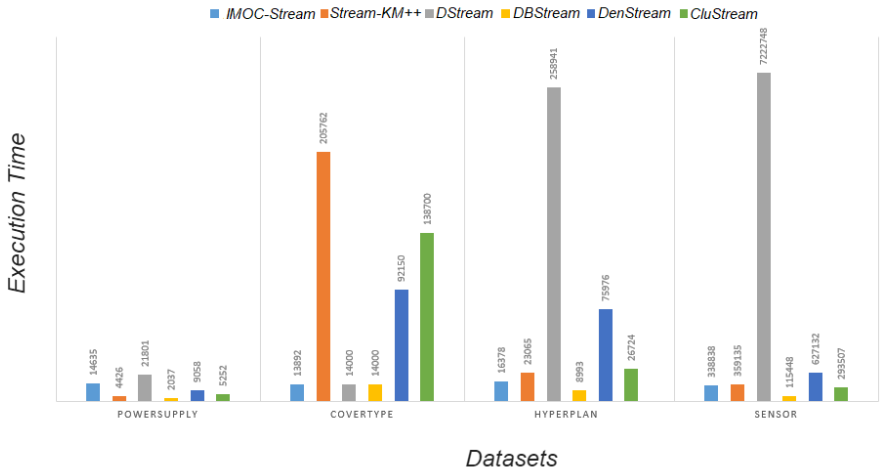


Figure 6.11 – *Execution time in milliseconds of each algorithm for every dataset.*

Figure 6.12 compares the time allocation of our method and the Ant-tree algorithm. We observe that IMOC-Stream requires less memory allocation than Ant-tree, on all the datasets. We note that these results are because Ant-tree stores all the data points, making the complexity approximately $n \times d$. While IMOC-Stream stores only the synopsis that is equal to $K \times d$, and when we add the other algorithms' solutions, the memory complexity becomes $\sum_{j=1}^{m} K_j \times d$, where m is the number of clustering solutions.

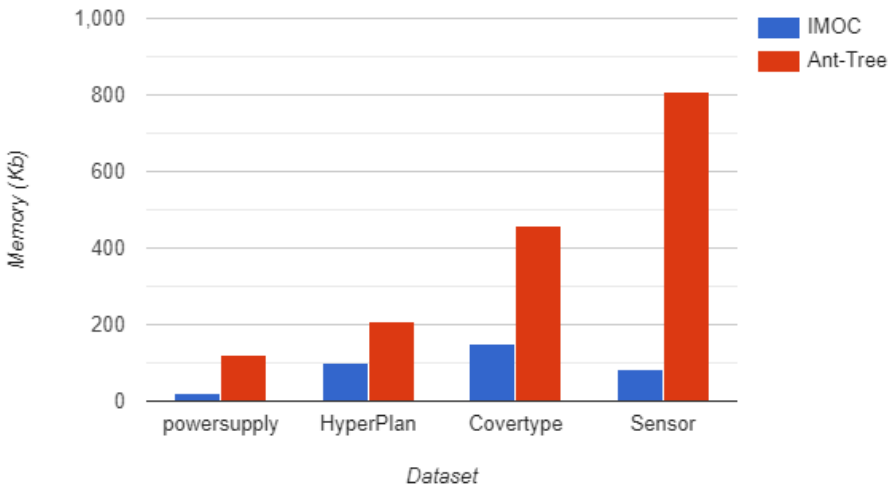


Figure 6.12 – *Memory allocation in Kilobyte of IMOC-Stream and Ant-tree for every dataset.*

6.5 Conclusion

This chapter presents a new method for clustering data stream based on a multi-objective algorithm called IMOC-Stream. Unlike those single-objective clustering techniques that have employed only one objective function, IMOC-Stream employs two objective functions to find clusters of arbitrary shaped clusters and enhance the clustering quality. IMOC-Stream uses a two-phase process: 1) online phase: creating several clustering solutions based on different algorithms and genetic operators 2) offline phase: construction of an optimal partition from the discovered clusters. We applied our method on large stream datasets and compared it to a different stream clustering algorithm. The experiments show the effectiveness of IMOC-Stream for detecting arbitrary shaped, compact, and well-separated clusters with better execution time.

7 Regions of Interests Selection in Histopathological Images using Subspace and Multi-Objective Stream Clustering

7.1 Introduction

Histopathology is a branch of histology that refers to the examination of diseased tissues or cells. Routine analysis of specimens is a common practice for cancer diagnosis, prognosis, and treatment. Firstly, The pathologist prepares the extracted biopsy samples by different processing techniques: fixation, embedding, portioning, and staining [Mescher, 2018]. Then, the preprocessed samples are analyzed under a microscope. Currently, biopsy samples are digitized by a whole digital scanner (WSD). The advantage of WSD compared to standard microscopes is their ability to scan the entire glass slides to produce digital slides denoted as whole slide images (WSI). The pathologist analyzes the WSI on the computer screen and performs diagnosis based on a specific software. Figure 7.1 highlights the difference between the WSI obtained by a WSD and the region of interest (ROI) obtained by a standard microscope. The pathologist manually selects these ROIs under the conventional microscope or from digitized WSIs.

The purpose of this chapter is to use clustering methods to overcome the related problem of non-relevant patches. The objective of clustering is to select the most relevant patches for classification instead of using all patches.
This chapter presents a Subspace Multi-Objective method for Patch Clustering and Selection (SMO-HPS) by combining subspace clustering and multi-objective techniques to overcome the problem of non-relevant patches. Our method's goal is to select the relevant patches for classification instead of using the whole image or all selected patches by the sliding window method. The rest of the chapter is organized as follows: in section 7.2, we present the art literature's relevant state. In section 7.3, we describe our method and its main features. In section 7.4, we present

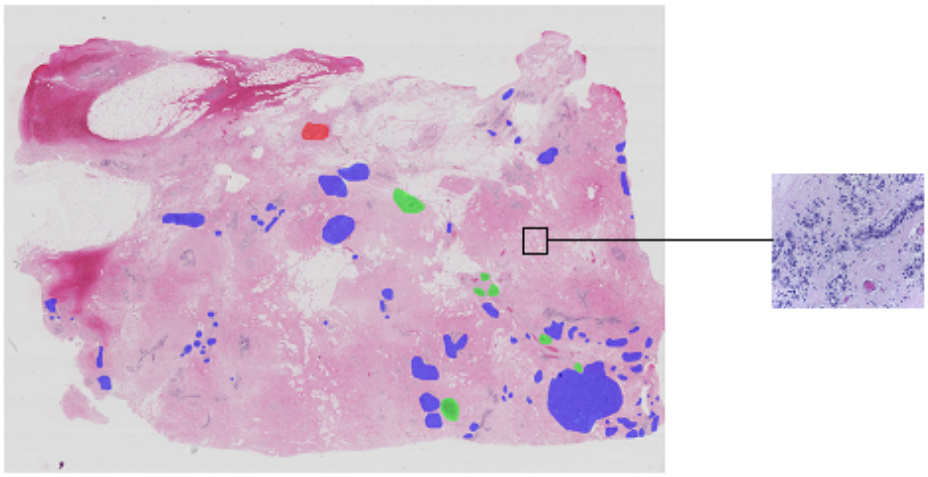


Figure 7.1 – *Example of a region of interest in a histopathological image.*

the results of the experiments and compare the method to some known clustering methods. Finally, we conclude this chapter.

7.2 Patch selection methods

Several attempts have been made in the literature to automate the classification of histopathological datasets [Jimenez-del Toro et al., 2017]. These datasets are composed of a set of digitized ROIs or WSIs. In general, the analysis WSIs require to extract relevant ROIs to classify them by Deep Learning methods. The extraction process is performed by various techniques such as clustering methods [Zhu et al., 2017].

Despite the advantages of the convolutional neural networks (CNN), these architectures are prone to overfitting small datasets. To solve this limitation, data augmentation techniques have been largely considered. The purpose of data augmentation techniques is to generate multiple target images from a source image by standard methods such as rotation, reflection, and patch selection [Krizhevsky et al., 2012]. For ImageNet classification, authors in [Krizhevsky et al., 2012] extracted 224×224 random patches from input images of size 256×256. Then, the generated patches have been augmented by reflections and rotations. These techniques increased the size of the training dataset by a factor of 2048.

Their high resolution characterizes histopathological images.i.e. 2048×1536 [Aresta et al., 2019]. In histopathological image analysis, the patch extraction method was largely considered. This technique extracts small patches from the high-resolution ROIs to adapt the input images' size to convolutional neural network inputs (generally 224×224). Moreover, it helps to generate big volumes of samples to overcome overfitting. The large difference between the source image and extracted patches can cause a loss of information in natural images because the patch frame should be centered around the object of interest. On the other hand, for histopathological images, the regular distribution of basic biological forms

allows the exploitation of small patches.

In histopathological image analysis, the patch extraction method extracts small patches from the high-resolution ROIs to adapt the input images' size to convolutional neural network inputs (generally 224×224 or 256×256). Several investigations proposed the exploitation of the sliding window technique. In this method, a small window is sliding from the left to the right across the image and from the top to the bottom to extract patches. For prediction, a majority voting process is performed to predict the class of the entire image. For instance, authors in [Xu et al., 2017] randomly extracted 256×256 from input images of size 3078×2752. This process generated approximately 28 000 patches from 4544 source images. In another investigation [Janowczyk and Madabhushi, 2016], each image of size 1388×1040 was divided into 32×32 patches with a stride S = 32. Authors in [Spanhol et al., 2016] used two strategies for patch extraction. The first uses a sliding window with an overlapping rate r = 0.5 to generate 260 32×32 patches, whereas the second selects 1000 random patches from each input image.

Despite the advantage of the sliding window technique in data augmentation, it can generate non-discriminant patches for classification. Moreover, the voting process is prone to non-relevant patches. To solve this limitation, a patches screening method that combines both the k-means clustering algorithm and CNN has been proposed to select discriminative patches [Spanhol et al., 2016].

7.3 Proposed method

This section introduces a new patch selection algorithm that combines subspace clustering and multi-objective techniques (SMO-HPS) to create a subspace multi-objective clustering algorithm that optimizes two objective functions. The purpose of using stream clustering in this chapter is to divide the image into windows of instances and process each one at a time, which is similar to stream processing. The method aims at improving the clustering quality and finds the relevant patches in a histopathological image. We describe in the following sections the main properties of our algorithm. Assuming that each patch contributes differently to each cluster's creation, we associate a different weight vector α for each feature with each cluster. A patch is associated with a weight vector α_b, which is the average weight of the patch's features. And a weight vector α_w, which is the current window's weight vector (a subset of instances). The weight $\alpha_{bi} + \alpha_{wj}$ describes the relevance of a patch (i, j). We assume that the data consists in a sequence $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ of n (potentially infinite) elements arriving at times $t_1, t_2, ..., t_n$, where $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$. Table 7.1 present the notations used in this chapter.

Notation	Description
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$	n number of instances
$\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$	d-dimensional data point
$\mathbf{w}_c = (w_c^1, w_c^2, \ldots, w_c^d)$	Center of cluster c
$\delta(c, r)$	Euclidean distance between clusters c and r
α	Matrix of feature weights in each cluster
K and K_{max}	Number and maximum number of clusters
λ	Adjustment parameters for feature weight
$\phi(\mathbf{x}_i)$	Assignment function (cluster corresponding to data point $\mathbf{x_i}$)
pH and pW	Height and Width of a patch
Window and Block	A window is a subset of instances and a block is a set of features A patch is an intersection between a window and a block
W and B	Number of windows and number of blocks
α_w and α_w	Weight of a window and weight of a block
$Intensity_p$	The intensity of a pixel p

Table 7.1 – *Notations used in SMO-HPS*

In this chapter, we use the Subspace Clustering (SC) technique to select patches. This choice is due to the capacity of SC to detect subsets of data that contribute the most to the clustering, which makes them relevant. Therefore, each patch is considered as a subspace. A subspace is a predefined length subset of features. Each feature is associated with a weight α to measure its relevance.

In our previous work [Attaoui et al., 2020], we proposed a cost function that needs to be minimized. The cost function optimizes four terms: ϕ is the assignment function of a point p to its closest cluster c, $\mathcal{W}$ is the set of prototypes or centers of the clusters, α and β are respectively the weights of the blocks and the features. This cost function is presented below for a window $\mathcal{X}^{(t+1)} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{t+1}\}$:

$$J^{(t+1)}(\phi, \mathcal{W}, \alpha, \beta) = \sum_{c \in C} \sum_{b=1}^{P} \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} \mathcal{K}^T \left(\delta(c, \phi(\mathbf{x}_i))\right) \alpha_c^b \mathcal{D}_{\beta_{cb}} + J_{cb} + I_c \tag{7.1}$$

where $\mathcal{D}_{\beta_{cb}} = \sum_{j=1}^{d_b} \beta_{cb}^j (x_i^j - \omega_c^j)^2$

The terms $I_c = \lambda \sum_{b=1}^{P} \alpha_c^b \log(\alpha_c^b)$ and $J_{cb} = \eta \sum_{j=1}^{d_b} \beta_{cb}^j \log(\beta_{cb}^j)$ respectively represent the weighted negative entropies associated with the subspaces weight vectors and the features weight vectors. The parameters λ and η are used to adjust the relative contributions made by the features and subspaces to the clustering. $\delta(c, \phi(\mathbf{x}_i))$ is the euclidean distance between a cluster center c and the center of the cluster $\mathbf{x}_i$ belongs to. $\mathcal{K}^T(\delta) = \mathcal{K}(\delta/T)$ is the neighborhood function, T controls the width of $\mathcal{K}$.

However, the method presented above is a single-objective method, and it does not perform well with histopathological image clustering. The algorithm presented in this chapter is a multi-objective method that optimizes two objective functions to obtain high-quality solutions. The single-

objective subspace algorithm's conversion to the multi-objective framework requires introducing some techniques and adapting the cost function. We describe these techniques in the following sections.

Adaptation of the cost function to the multi-objective framework

One of the important aspects of MOC is the choice of suitable objective functions to be optimized simultaneously. For each clustering solution, several quality measures exist, and the goal is to maximize inter-cluster similarity and minimize intra-cluster similarity. To satisfy these requirements, we introduce a model with two cost functions corresponding to two objective functions. J_{within} is the within-cluster dispersion, and it maximizes the compactness of a cluster. J_{sep} is the combination of the separateness between clusters and the weighted negative entropy that needs to be minimized. The cost functions optimize three parameters: ϕ the assignment function, $\mathcal{W}$ the prototypes set and α the weight of each feature in a cluster c. For a data window $\mathcal{X}^{(t+1)} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{t+1}\}$, and a clusters set C, the objective functions are calculated as follows:

$$J_{within}^{(t+1)}(\phi, \mathcal{W}, \alpha) = \sum_{c \in C} \sum_{\mathbf{x}_i \in \mathcal{X}^{(t+1)}} (\delta(c, \phi(\mathbf{x}_i))) \sum_{j=1}^{d} \alpha_c^j (x_i^j - \omega_c^j)^2, \tag{7.2}$$

where $\sum_{j=1}^{d} \alpha_c^j = 1$, $\delta(c, \phi(\mathbf{x}_i))$ is the shortest path between a cluster center c and the center of the cluster $\mathbf{x}_i$ belongs to. α_c^j is the weight of j^{th} feature in cluster c, d is the number of features (width of the image).

$$J_{sep}^{(t+1)}(\phi, \mathcal{W}, \alpha) = \sum_{c \in C} (\overline{\alpha_c} / (Separateness_c + \epsilon)) + I_c \tag{7.3}$$

$$I_c = \lambda \sum_{j=1}^{d} \alpha_c^j \log(\alpha_c^j)$$

$$\overline{\alpha_c} = \sum_{j=1}^{d} \gamma_j \alpha_c^j / \sum_{j=1}^{d} \gamma_j; \;\; \gamma_j = 1 \text{ if } \alpha_c^j > \frac{1}{D} \; else \; 0$$

$$Separateness_c = \sum_{p \in C} \sum_{j=1}^{d} (\omega_c^j - \omega_p^j)^2$$

$$\text{Where: } \gamma_j = \begin{cases} 1 \text{ if } \alpha_c^j > \frac{1}{D} \\ 0 \; else \end{cases}$$

I_c is the weighted negative entropy, $\overline{\alpha_c}$ is the average of weights α, and $Separateness_c$ is the sum of the euclidean distances between the actual cluster and the other clusters, ϵ is a pre-defined value that prevents the denominator from becoming zero.

Encoding and initialization scheme

In data stream clustering, data points can't be stored and processed in one pass. Therefore, we chose a new genetic representation called centroid-based representation to facilitate the clustering update as the data flow.

The cluster center is represented by an array of d real numbers describing its coordinates, where d is the dataset's dimension (number of features). After each iteration, only the centroid is updated using the points assigned to it.

The initialization process is the first and the most crucial step in the multi-objective methods as a good scheme can lead to faster convergence, while a bad scheme can lead to wrong final solutions. In most multi-objective clustering methods, the clustering solutions in the initial population have been generated randomly. This random initialization depends on the encoding used in the algorithm (Cluster centers, assignment). In this chapter, we initialize solutions using the K-means algorithm [MacQueen et al., 1967] with different K values. K-means clustering algorithm is an efficient partitioning algorithm. However, in most implementations of the K-means algorithm, the first K centroids are chosen randomly. This initialization can lead to either slow convergence or bad results. This chapter uses a different K-means initialization method based on the distances between the initial centroids to have well-separated clusters. First, we randomly chose the first point from the dataset as a centroid. The second center is also selected randomly, but the probability of selecting a case is proportional to the distance (square euclidean) of it to the first centroid. The third centroid is also chosen randomly with the probability of selection proportional to the distance of a case to the nearest of those two centers, and so on till we reach K centers.

The primary objective of this type of initialization is to replace the random initial population by good clustering solutions generated by K-means. The weights α for each feature in each cluster center are initialized randomly with respect to $\sum_{j=1}^{d} \alpha_c^j = 1$ and $0 \leqslant \alpha_c^j \leqslant 1$.

Genetic functions

Genetic operators are essential for MOC methods as they enable the variety and diversity of the clustering solutions. In our method, we use One Point Crossover and Random Resetting Mutation described below.

Single point crossover: a crossover point i chosen randomly, and the resulted chromosome is composed of values from the beginning to i of the first parent and from i+1 to the end of the second parent. The crossover point is chosen based on the smaller K between the two solutions.

Random resetting mutation: gives a random value from a pre-defined range to the randomly selected position. The two genetic functions are illustrated in Figure 7.2.

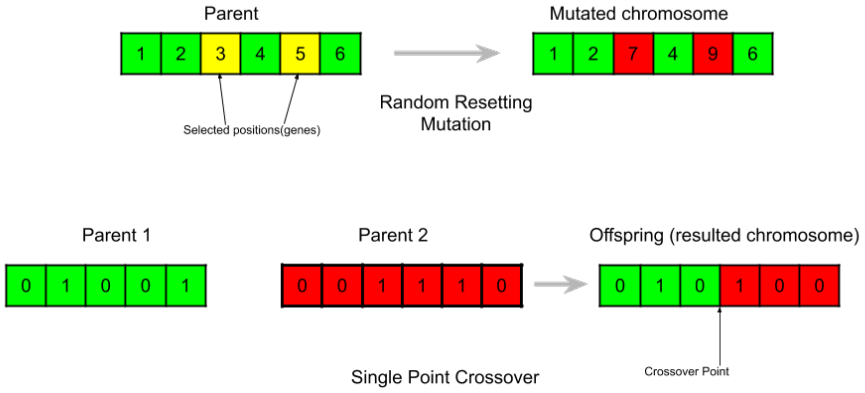


Figure 7.2 – *Single Point Crossover and Random Resetting Mutation.*

We must note that if two solutions have different number of clusters, we can still apply both genetic functions as shown in Figure 7.2. We apply the mutation operator on each solution in the pareto set, while the crossover is applied on each two solutions in the pareto-set.

Update clustering solutions

In the online phase, clustering solutions generated from the previous windows need to be updated with new points from the current window. For each clustering solution generated before, we assign each data point in the current window to the closest cluster then the prototype $\omega_c^{(t)}$ is updated as follows:

$$\omega_c^{(t+1)} = \frac{\omega_c^{(t)} n_c^{(t)} \gamma + \mathbf{z}_c^{(t)} m_c^{(t)}}{n_c^{(t)} \gamma + m_c^{(t)}} \tag{7.4}$$

Where $\omega_c^{(t)}$ is the previous prototype, $n_c^{(t)}$ is the number of points assigned to the cluster, $\mathbf{z}_c^{(t)}$ is the new prototype computed from the current window. $m_c^{(t)}$ is the number of points assigned to the cluster c in the current window: $n_c^{(t+1)} = n_c^{(t)} + m_c^{(t)}$. γ is the decay factor that decreases over time to give more importance to most recent data $0 < \gamma < 1$ with $\gamma = 1$ all data will be used from the beginning; with $\gamma = 0$ only the most recent data will be used.
The α weights are updated as follows:

$$\alpha_c^j = \frac{exp(D_c^j)}{\sum_{i=1}^{d} exp(D_c^i)} \tag{7.5}$$

$$D_c^j = \sum_{p \in C} |\omega_c^j - \omega_p^j|$$

Where: D_c is the sum of euclidean distances between the cluster c and the other clusters in the population.

Leader selection

Optimizing multiple clustering objectives ought to produce a set of non-dominated solutions. The most suitable solution that indicates the final clustering result should be retrieved with an expert's help. However, some other selection methods exist. In our method, we used an internal quality measure called Davies Bouldin index [Davies and Bouldin, 1979] to select the best solution among the Pareto-set. The choice of an internal index is because the data might not be labeled. We sort all the solutions by their fitness (internal measures values), and we choose the best one as an output of the algorithm. Davies Bouldin index helps identify sets of clusters that are compact and well separated. The Davies-Bouldin index is presented in Section 3.2.1.

Adaptation to RGB data type

Most of the clustering algorithms, including our method, are designed to deal with numerical data since most of the datasets available contain numerical data. However, in histopathological datasets, the data represents the values of pixels in RGB. The algorithm needs to be adapted to this kind of data. To do so, we use the following steps:

- **Conversion of RGB to CIE-L*ab**: Since distance can't be computed in RGB space, we convert all the values to CIE-L*ab, which is a color space defined by the International Commission on Illumination (CIE) in 1976. It expresses color as three values: L* for the lightness from black (0) to white (100), a* from green (-) to red (+), and b* from blue (-) to yellow (+).
- **Adaptation of the euclidean distance**: the euclidean is a measure of similarity between two points, it is mostly used for numerical data. In this chapter, we present a euclidean distance to compute the similarity between RGB data points. The Euclidean distance between two pixels $p_1 = (L_1, a_1, B_1)$ and $p_2 = (L_2, a_2, B_2)$ is presented below:

$$Euclidean_{LaB} = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (B_1 - B_2)^2} \tag{7.6}$$

- **Encoding**: We modify the encoding of centroids presented in the section above to handle RGB values. Centroids are represented by arrays of $d \times 3$ where d is the dimension of the data and each point have 3 values (R, G, B) representing the red, green and blue respectively.

Relevant Patch Selection

We assume a histopathological image is an array of $n \times d$, each value (i, j) represents a pixel. A patch is a subset of the image of size $pH \times pW$ where pW and pH are the patch's width and height. We divide the image on windows of pH instances, and we process each window at a time. The features also are divided on blocks of pW features. We denote W and B the number of windows and the number of blocks, respectively.

The feature weight matrix α is a $K \times d$ where K is the number of clusters. Each value α_c^j represent the weight of the jth feature in the cth cluster. This matrix is updated for each window as described above. At the end of each window, we compute a new weight α_{wi} for the current window. This value is used to evaluate the window and it is calculated as follows:

$$\alpha_{wi} = \frac{1}{d} \sum_{j=1}^{d} Intensity_j . \frac{1}{K} \sum_{c \in C} \alpha_c^j, \tag{7.7}$$

where $Intensity_j$ is the sum of pixels px_{ij}'s intensities for the feature j and $0 < i < N$ where N is the size of the patch. We use the intensity as a weight here, assuming that the pixels with high intensity gives more information. The intensity of a feature j is calculated as follows:

$$Intensity_j = \sum_{i=1}^{N} Max(L_i, a_i, B_i) \tag{7.8}$$

At the end of the process, we calculate the weight of each block α_{bi} from the matrix α as follows:

$$\alpha_{bi} = \frac{1}{pW} \sum_{j \in features_{bi}} Intensity_j . \frac{1}{K} \sum_{c \in C} \alpha_c^j, \tag{7.9}$$

where $features_{bi}$ is the subset of features belonging to block b_i.

Weight α_w is a vector of W containing the weight of each window, and α_b is a vector of B containing the weight of each block. Each patch (i, j) is associated with a weight (α_{wi} + α_{bj}). We sort all the patches based on their weight ans we select a percentage of relevant patches. The rate of selection is chosen based on the expirement in Section 7.4. The indices of a patch (i, j) in the original image are: $[(i \times pH, i \times pH + pH), (j \times pW, j \times pW + pW)]$. Figure 7.3 illustrates the process of patch extraction.

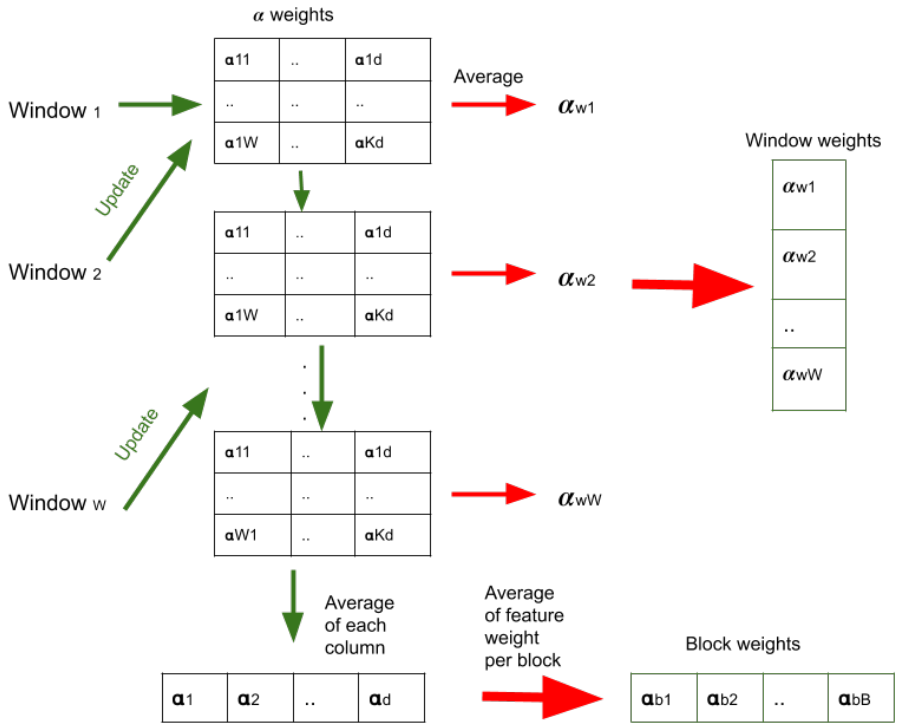


Figure 7.3 – *Process of extracting relevant patches. Window weights and Block weights are the final weight vectors used to determine the extracted features.*

Subspace Multi-Objective Histopathological Patch Selection algorithm

The selection of ROIs in histopathological images is a crucial technique to save time and memory; instead of processing the whole image, patch selection provides subsets that give the same information about the disease as the entire image. This chapter proposes a new method of patch selection based on subspace stream clustering combined with multi-objective techniques. The method starts with initializing many clustering solutions using different initialization of the K-means algorithm. It assigns random weights to each patch. In each step of the algorithm, new clustering solutions are created using Mutation and Crossover operators. The objective functions are computed for each solution, and only the Pareto-optimal solutions are stored in the Pareto-set. The weights α are updated using the solutions in the Pareto-set for each window of data. At the end of the algorithm, the best solution from the Pareto-set is selected according to Section 7.3. The relevant patches are extracted according to Section 7.3.

We proposed another version of our algorithm to deal with numerical data. This version have the same steps as the RGB version except for the patch selection step. The data is seen as a stream of instances, each instance is processed at a time and the solutions are updated as described in the previous sections. At the end of the process, the best solutions is selected. The algorithm proposed in this chapter is presented in Algorithm 12.

Algorithm 12 Subspace Multi-Objective Histopathological Patch Selection algorithm

Result : Optimal clustering solution, Set of relevant ROIs

From the first window: generate several clustering solutions using K-means with different K;
Encode the clustering solutions following scheme in Figure 7.3;
Apply Crossover and Mutation following Figure 7.2. Add new solutions to the population of chromosomes;
Compute objective functions following Equations (7.2) and (7.3). Store non-dominated solutions in the Pareto-set;

while *There is data available* **do**
    For each clustering solution in the pareto-set, assign each point to the closest cluster;
    Update each cluster in each clustering solution using the new points assigned as described in Equation (7.4);
    Update weights α following Equation (7.5);
    Apply Crossover and Mutation following Section 7.3. Add new solutions to the population of chromosomes;
    Compute objective functions of the clustering solutions in the pareto-set and the new solutions generated. Update the pareto-set with the new non-dominated solutions;
    Calculate Window weight according to Equation (7.7);

end

Calculate blocks weights according to Equation (7.9);
Extract relevant patches based on block and window weights;
Select best solution among the pareto-set solutions as described in section 7.3

7.4 Experimental results

To show the effectiveness of our method on both clustering and patch selection, we developed two versions of SMO-HPS. The first one is a stream method that processes numerical datasets. The second version deals with the RGB data type, and it is suitable for histopathological datasets. Assuming large-dimensional data arrives as a continuous stream, SMO-HPS divides the data into windows and continuously processes each window. The window size depends on the available memory and the original dataset's size. The size of the window for each dataset is shown in Table 7.2. We set the *maxK* parameter for the maximum clusters with K-means clustering to 15. We set the time interval between two windows to 1 millisecond. We present the results of SMO-HPS on synthetic and histopathological datasets in the following sections.

Synthetic Datasets

We compared SMO-HPS to four well known stream algorithms: *StreamKM++* [de Andrade Silva and Hruschka, 2011], *DStream* [Tu and Chen, 2009], *DBStream* [Hahsler and Bolaños, 2016], *DenStream*

[Cao et al., 2006] and *CluStream* [Aggarwal et al., 2003] from R package streamMOA[1]. We describe the synthetic datasets in Table 7.2.

Dataset	Instances	Features	Classes	$\|Window\|$
1CDT	16000	2	2	100
1CHT	16000	2	1	100
4CE1CF	173251	2	2	1000
GEARS-2C-2D	200000	2	2	10000

Table 7.2 – *Description of the datasets used in experimentation*

The results of SMO-HPS on the synthetic datasets described above compared to the different algorithms based on Normalized Mutual Information (NMI) [Strehl and Ghosh, 2002] and Adjusted Rand index (ARAND) [Hubert and Arabie, 1985] are reported in Table 7.3. We note that these results are the average of ten different runs with each algorithm.

It is noticeable that SMO-HPS yields better results than all the other methods. These results are because our method uses subspace clustering to make relevant features contribute more to the clustering. While optimizing two objective functions to maximize intra-cluster similarity and minimize inter-cluster similarity at the same time. The use of genetic operators is essential to combine the best solutions and explore the potential local solutions.

Dataset	Metrics	SMO_HPS	StreamKM++	DStream	DBStream	DenStream	CluStream
1CDT	NMI	**0.900 ± 0.13**	0.719 ± 0.03	0.691 ± 0.10	0.631 ± 0.28	0.208 ± 0.05	0.621 ± 0.06
	ARAND	**0.939 ± 0.25**	0.662 ± 0.02	0.667 ± 0.13	0.610 ± 0.30	0.086 ± 0.04	0.583 ± 0.09
1CHT	NMI	0.785 ± 0.02	**0.876 ± 0.01**	0.489 ± 0.02	0.137 ± 0.03	0.258 ± 0.03	0.743 ± 0.03
	ARAND	0.834 ± 0.04	**0.920 ± 0.02**	0.415 ± 0.03	0.007 ± 0.025	0.145 ± 0.004	0.743 ± 0.023
4CE1CF	NMI	**0.614 ±0.03**	0.542 ± 0.12	0.597 ± 0.17	0.553 ± 0.02	0.071 ± 0.05	0.409 ± 0.10
	ARAND	**0.491 ± 0.06**	0.349 ± 0.17	0.460 ± 0.19	0.382 ± 0.07	0.001 ± 0.07	0.291 ± 0.11
GEARS_2C_2D	NMI	**0.732 ± 0.01**	0.543 ± 0.03	0.160 ± 0.12	0.001 ± 0.00	0.021 ± 0.02	0.301 ± 0.02
	ARAND	**0.797 ± 0.01**	0.449 ± 0.03	0.154 ± 0.16	0.001 ± 0.00	0.010 ± 0.01	0.219 ± 0.02

Table 7.3 – *NMI and ARAND results of SMO-HPS on synthetic datasets. Value after ± is the standard deviation and the value in bold is the best value.*

Patch Selection

For the histopathological images, we use the RGB version of the SMO-HPS algorithm described in section 7.3. We set pW and pH of the patches to 224. The extracted patches will have the size 224×224 which is suitable for most of the deep learning methods. Each window will have 224 from the image, a clustering will be produced for each window and the α_w weight will be calculated as described in the previous sections.

The overlapping rate is the percentage of pixels that are kept from the previous patch and re-used for learning with the new window. We set the value of the overlapping rate to 0.3 and $maxK$ to 15. The datasets are described in Table 7.4.

[1]https://github.com/mhahsler/streamMOA

Dataset	Resolution (px)	Nb of Images	Classes	$\|Patch\|$
Breakhis	700 x 460	7909	2	224 x 224
Lymphoma	1388 x 1040	375	3	224 x 224
ICIAR 2018-A	2048 x 1536	400	4	224 x 224
MITOS-Atypia	1539 x 1376	1188	3	224 x 224

Table 7.4 – *Description of the histopathological datasets used in this experimentation*

The breast cancer dataset (Breakhis) [Spanhol et al., 2015] was collected from 82 patients and digitized under various magnifications (40×, 100×, 200×, 400×). This dataset is composed of two main classes: benign and malignant. Each class is categorized into four additional subclasses: adenosis (A), fibroadenoma (F), phyllodes tumor (PT), tubular adenona (TA), and ductal carcinoma (DC), lobular carcinoma (LC), mucinous carcinoma (MC) and papillary carcinoma (PC). In our experiments, we used the binary version of the 40× magnification.

The lymphoma dataset [Shamir et al., 2008] classifies the non-hodgkin's lymphomas into three categories: chronic lymphocytic leukemia (CLL), follicular lymphoma (FL), and mantle cell lymphoma (MCL). To prepare this dataset, 30 slides have been digitized by the Zeiss Axioscope light microscope.

The ICIAR 2018-A[2] dataset was proposed in the ICIAR 2018 challenge. In this dataset, the beast histology images have been digitized under 200x magnification and annotated into four classes: normal, benign, in situ carcinoma, and invasive carcinoma.

The MITOS-Atypia dataset[3] is used for mitosis detection and nuclear atypia scoring (NAS) on invasive breast carcinoma slides. In this investigation, we used the NAS version, where the different slides have been annotated into low-grade atypia, moderate grade atypia, and high-grade atypia. Despite the previous researches, we treated the NAS as a classification task.

To show the effectiveness of our method in extracting relevant patches, we compared the extracted ROI's to those obtained by a standard sliding window method that extract all the possible (224 × 224) patches by creating a window that moves through the features and the instances by a 224 step with an overlapping rate r=0.7. We set the parameter λ to 7.

We select a percentage of relevant patches from each image. First, the patches are sorted based on their weight, which is the sum of the window weight, and the block weight described in the previous section. Then, we select a percentage of relevant patches. This rate depends on the image. We try different rates for each image, and we choose the one that gives the best DBI value. Figure 7.4 presents an example of rate selection for an image from Breakhis dataset. In this figure, the rate that gives the best DBI value is 0.3.

[2]https://iciar2018-challenge.grand-challenge.org/Dataset/
[3]https://mitos-atypia-14.grand-challenge.org/Dataset/

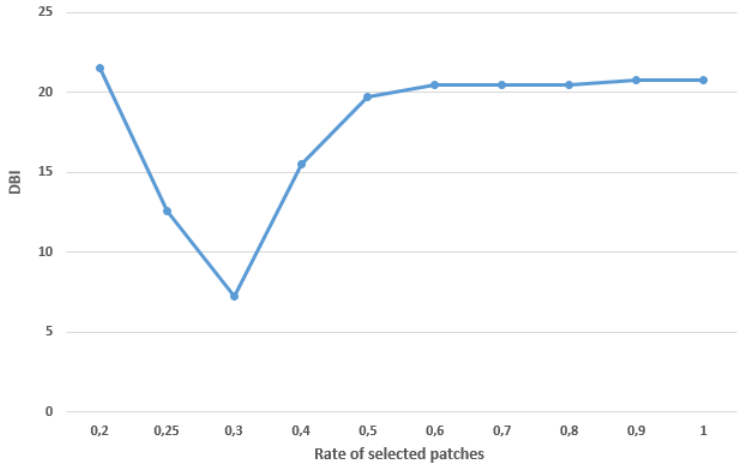


Figure 7.4 – *Example of rate selection for an image from Breakhis dataset. The rate that gives the best DBI value is* 0.3.

Table 7.5 shows, for each dataset, the number of patches extracted by a sliding window method compared to the number of patches extracted by SMO-HPS. Figure 7.5 shows examples of patch selection from each class of the Breakhis dataset compared to patches obtained by a sliding window method. We notice that our approach gives a significantly lower number of relevant patches. As we see in class ductal-carcinoma, the sliding window method extracted redundant patches while our approach omitted them. These redundant patches can affect the computational time without adding information. The same comment can be made for class papillary-carcinoma, which is where the sliding window method extracted patches with their background, shows that these patches contain almost no information. Our approach manages to extract only the patches with the most information. In the following sections, we show that these extracted patches can provide better learning performance with a lower computation time compared to learning on patches obtained by a sliding window method. To illustrates how these extracted patches are more relevant and represent better the histopathological image, we calculate Davies Bouldin Index on the entire dataset, and on only the extracted patches, the results are represented in Table 7.6. We also tested the K-means algorithm on the extracted patches and on the sliding window's extracted patches. We notice that the extracted patches' results are significantly lower than those on the entire dataset because the weights of these extracted patches are higher than the other patches, which means that they contributed more in the clustering process. We should note that the DB index needs to be

Dataset	Number of Patches extracted by SMO-HPS	Number of Possible Patches
Breakhis	**6018**	15960
Lymphoma	**15791**	17952
ICIAR 2018-A	**23194**	47970
MITOS-Atypia	**68923**	85536

Table 7.5 – *Number of patches extracted by a sliding window method compared to the number of patches extracted by SMO-HPS for each dataset.*

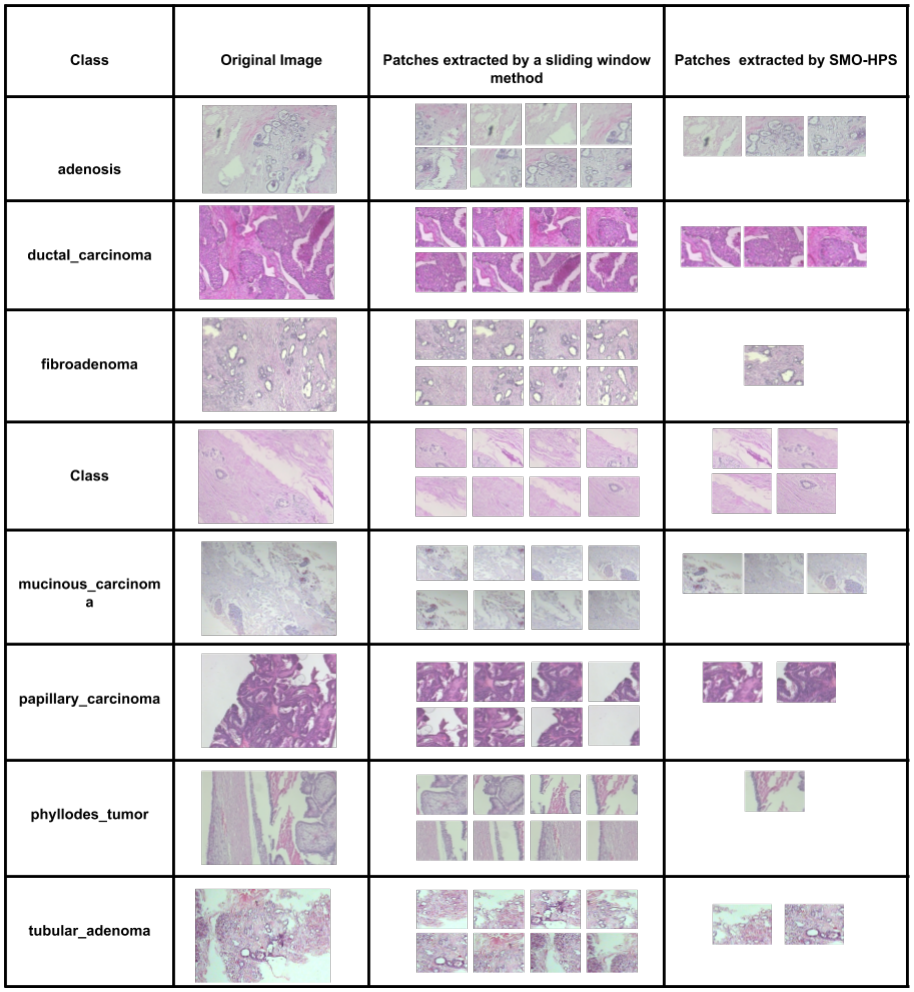


Figure 7.5 – *Examples of patch selection from each class of the Breakhis dataset compared to patches extracted by a sliding window method*

minimized. This patch selection also avoids the non-relevant regions of the image being diagnosed since they can give false information or contain no crucial information; in both cases, it isn't significant to process them. Adding to that, this clustering process can be done way faster than clustering the whole image. These results can help scientists make diagnoses on a smaller amount of data with the same results faster.

Dataset	DBI on Patches extracted by SMO-HPS	DBI on all the patches
Breakhis	**16707**	216225
Lymphoma	**23627**	36115
ICIAR 2018-A	**45127**	90303
MITOS-Atypia	**266214**	380159

Table 7.6 – *Average Davies Bouldin Index of SMO-HPS on the extracted subset of patches and on the whole set of patches. The value in Bold represent the best value.*

Dataset	K-means on Patches extracted by SMO-HPS	K-means on all the patches
Breakhis	**68126**	205498
Lymphoma	**129092**	143131
ICIAR 2018-A	**234689**	323789
MITOS-Atypia	**649141**	967778

Table 7.7 – *Average Davies Bouldin Index of K-means on the extracted subset of patches and on the whole set of patches. The value in Bold represent the best value.*

Patch Clustering

To transform the unstructured data (patches) into structured raw data, we used the InceptionV3 [Szegedy et al., 2016] pretrained network on ImageNet for feature extraction. Each input image is propagated in the network through inception blocks, then, the resulting 3D matrix from the last pooling layer is transformed to 1D vector which represents the extracted features. The result of feature extraction from all images is represented by a 2D matrix as:

$$X = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1m} & l_2 \\ x_{21} & x_{22} & ... & x_{2m} & l_c \\ ... & ... & ... & ... & ... \\ x_{k1} & x_{k2} & x_{k3} & x_{k4} & l_1 \end{bmatrix}, l_i \in \{l_1, l_2, ..., l_c\}, i \in [1, c] \tag{7.10}$$

where c is the number of categories, m is the number of features, and k is the number of instances or images.

To illustrate our method's clustering performance, we perform the first version of our method on the resulted structured raw dataset. Each instance of this dataset is a vector representing a patch. Figure 7.6 shows an example of clusters found by Kmeans with K = 2 and our method on the Breakhis dataset. We noticed that the similarity between the patches in the same cluster is higher for SMO-HPS compared to K-means. This is since our approach, unlike K-means, is an automatic K determination method, meaning that it finds the right number of clusters. Our method manages

to find a relevant subset of patches that represent better the whole dataset while providing a proper partitioning for these patches. This experiment aims at providing groups of similar tissues to help researchers detect cancer-based on a group of regions of interest instead of analyzing the whole set of patches, therefore, minimizing the computational time.

Medical Significance

The presented clusters by SMO-HPS shows a structure with similar morphology. We also observe some representative tissues. After consulting an Anatomo-pathologist, we discovered the significance of some clusters. Cluster 1 represents Fibrous connective tissues; the ones in Cluster 2 represent adipose tissues, and the ones in the third cluster represent structures with a large distribution of nuclei. Finally, Cluster 5 represents Hyaline Cartilage tissues.

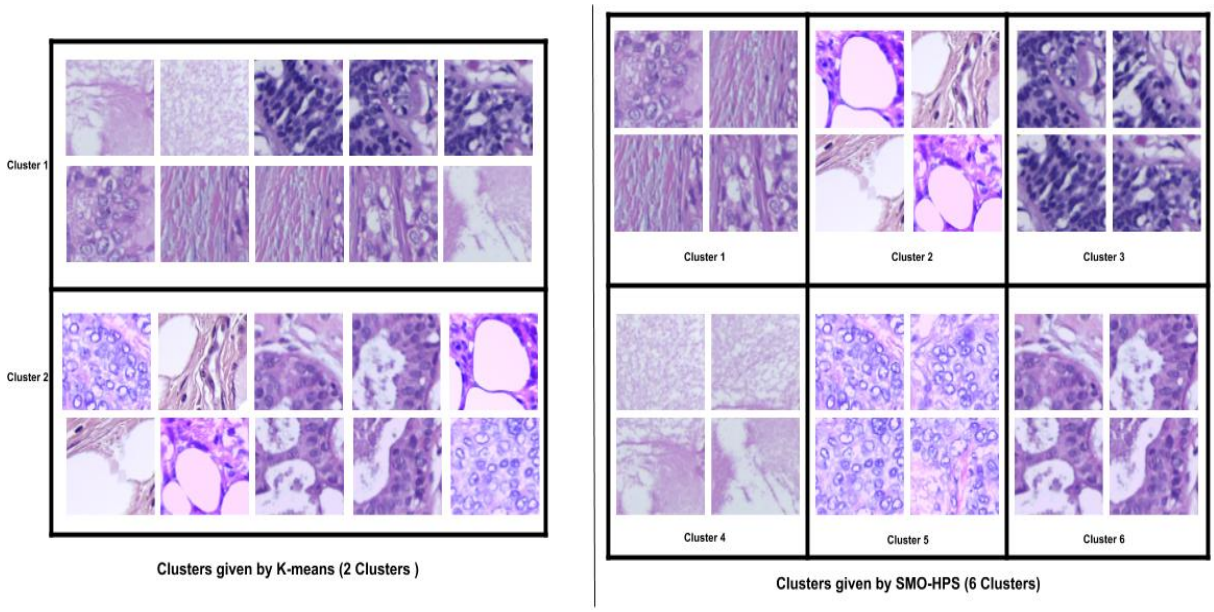


Figure 7.6 – *Example of clusters given by K-means algorithms and clusters given by SMO-HPS*

Figure 7.7 compares Davies Bouldin index values for SMO-HPS to the K-means algorithm on the datasets mentioned above. We notice that our method yields lower values on all the datasets compared to K-means. We note that the Davies Bouldin index needs to be minimized, which means that our method performs better than K-means while extracting relevant features.

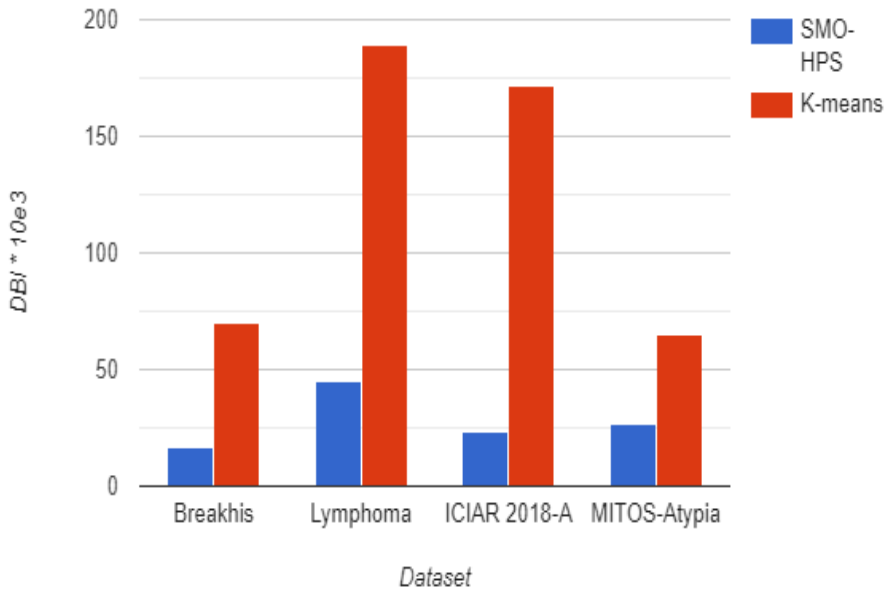


Figure 7.7 – *Comparison between DBI value of SMO-HPS and K-means with K = 2 on patches extracted by SMO-HPS.*

Classification using Patches

To show the relevance of the extracted patches by SMO-HPS, we trained the convolutional neural networks (InceptionV3 and VGG16) on the extracted patches by SMO-HPS and by the sliding window method. We settled for the breakhis, lymphoma, and ICAR-2018-A datasets to validate the proposed method due to space limitations.

For training, we fine-tuned the pre-trained ImageNet's InceptionV3 [Szegedy et al., 2016] and VGG16 [Simonyan and Zisserman, 2014] models on the target histopathological datasets. We fine-tuned the last five trainable layers in VGG16 and the final trainable 44 layers in InceptionV3. We trained the models in 30 epochs with a window size of 64. We used the Nesterov's Accelerated Momentum (NAG) with adaptive learning. The learning rate was initialized to 0.01, decay to 10^{-6}, and momentum to 0.9. We used the stratified hold out method for evaluation: 70% for training and 30% for testing. For training, we evaluated each patch separately. For testing, we performed unweighted voting between patches to predict the class of the whole HPF. We conducted our experiments on an NVIDIA GeForce GTX 1060 GPU running on a PC with a CPU (Intel Corei5) and 8 GB RAM.

Tables 7.8 and 7.9 present the obtained results by the Inception and the VGG16 networks, respectively. The trained network on the extracted patches by SMO-HPS was tested on the test set generated by SMO-HPS and the test set generated by the sliding window method. We performed the same process on the trained model on the sliding window training data.

Dataset	Test-set/Trained-model	Sliding Window	SMO-HPS
Breakhis	Sliding Window	95.82%	91.65%
	SMO-HPS	96.32%	91.48%
ICIAR-2018-A	Sliding Window	72.72%	75.75 %
	SMO-HPS	69.69 %	69.69 %
Lymphoma	Sliding Window	91.15 %	91.15 %
	SMO-HPS	86.72 %	89.38%

Table 7.8 – *Results of the Inception network*

Dataset	Test-set/Trained-model	Sliding Window	SMO-HPS
Breakhis	Sliding Window	96.32%	93.48%
	SMO-HPS	96.49%	94%
ICIAR-2018-A	Sliding Window	72.72 %	66.66 %
	SMO-HPS	69.69 %	74.24%
Lymphoma	Sliding Window	89.38 %	91.15 %
	SMO-HPS	86.72 %	84.95%

Table 7.9 – *Results of the VGG16 network*

For the Breakhis dataset, training on the sliding window showed better results: InceptionV3 (96.32 %) and VGG16 (96.49%), this is because it provides a large amount of data for the network and ensures generalization. However, testing the network on patches extracted by SMO-HPS yields better results. It prevents the weak decision generated from non-relevant patches.

Despite the Breakhis dataset, the best results on ICIAR-2018-A and Lymphoma datasets have been obtained by the trained model on the SMO-HPS data and tested on the sliding window data (75.75 % and 91.15 %) based on the Inception and the VGG16 networks, respectively. The number of extracted patches by SMO-HPS for these datasets was more interesting than Beakhis, which provided more generalization for the trained model.

Finally, the best results on the ICIAR-2018-A dataset based on the VGG16 network have been obtained by the trained and tested model on the SMO-HPS data (74.24 %).

To resume, the extracted patches by SMO-HPS represent useful histopathological source datasets for training and testing. This technique selects relevant patches to generate efficient models when trained on the SMO-HPS data. It also helps in the prediction step, where we observed interesting results on the test sets when discarding non-relevant patches. We should note that, in some cases, this method should be combined with the sliding window to achieve more impressive results compared to their separate use.

Finally, we performed a comparative study between the obtained results and the other state-of-the-art achievements on the ICAR-2018, Breakhis, and the lymphoma datasets. For comparison, we selected only the deep learning methods. We observed that the architectures VGG16, Inception, and ResNet have been largely conceded for histopathological images classification. Moreover, the fine-tuning was more exploited than

training from scratch, which validates this technique's efficiency. For the ICIAR-2018 dataset, the best results have been achieved by the AlexNet [Nawaz et al., 2018] architecture when fine-tuning the last fully connected layers. On the other hand, for the Breakhis dataset, the proposed technique yields the best results when testing the VGG16 model on the selected patches by the SMO-HPS method. Finally, for the lymphoma dataset, the best results have been obtained by the DenseNet architecture [Nanni et al., 2019] when fine-tuning the last layers. Overall, the proposed method ranks near the top of the current state-of-the-art classification based models and achieved the best results on the Breakhis dataset.

Reference	Architecture	Fine tuning	Accuracy (%)
[Kaymak et al., 2017]	Back Propagation Neural Network (BPPN)	-	70.4
[Ferreira et al., 2018]	Inception-Resnet-V2	Fine tuning the last fully connected layers	76
[Nawaz et al., 2018]	AlexNet	Fine tuning the last fully connected layers	81.25
Our method	SMO-HPS + VGG16	Fine tuning the last 5 trainable layers	74.24
	SMO-HPS + InceptionV3	Fine tuning rhe last 44 trainable layers	75.75

Table 7.10 – *The obtained results on the ICAR-2018 dataset.*

Reference	Architecture	Fine tuning	Accuracy (%)
[Benhammou et al., 2018]	InceptionV3	-	82.7 $\pm$ 90.2
[Sun and Binder, 2017]	ResNet50	Fine tuning all layers	86.24 $\pm$ 3.44
[Zhi et al., 2017]	VGGNet16	Fine tuning the fully connected layers	89.12
[Zhi et al., 2017]	AlexNet	Fine tuning the last 3 fully connected layers	90.96 $\pm$ 1.59
Our method	SMO-HPS + VGG16	Fine tuning the last 5 trainable layers	96.49
	SMO-HPS + InceptionV3	Fine tuning the last 44 trainable layers	96.32

Table 7.11 – *The obtained results on the Breakhis dataset.*

Reference	Architecture	Fine tuning	Accuracy (%)
[Nanni et al., 2020]	Ensemble of Deeplabv3 with ResNet50	Fine tuning last layers	79.7
[Maguolo et al., 2019]	Ensemble of VGG16	-	85.87
[Janowczyk and Madabhushi, 2016]	AlexNet (Cifar-10 version)	-	92.00
[Nanni et al., 2019]	ResNet50	Fine tuning last layers	93.60
	DenseNet	Fine tuning last layers	96.58
Our method	SMO-HPS + VGG16	Fine tuning the last 5 trainable layers	91.15
	SMO-HPS + InceptionV3	Fine tuning the last 44 trainable layers	91.15

Table 7.12 – *The obtained results on the lymphoma dataset.*

Computational Time

Figure 7.8 shows the execution time of SMO-HPS on the extracted patches compared to the execution time on the whole set of patches. The time of the patch extraction process is added to the processing time. We can notice that the execution time on the extracted patches is shorter. This is due to the small amount of data compared to the sliding window method. Our method manages to extract relevant patches to improve the performance and reduce the computational time. This time reduction can also be beneficial to anatomo-pathologists since they can analyze less amount of data.

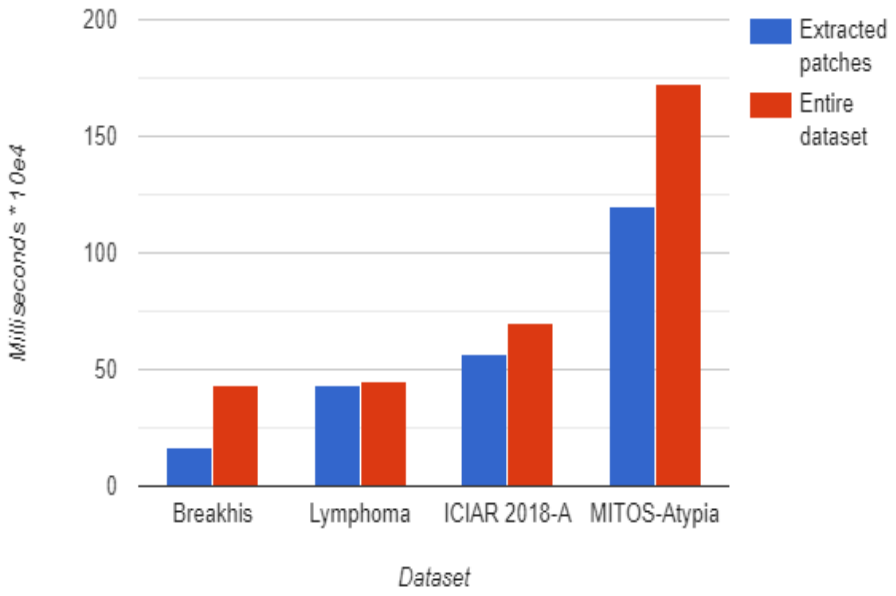


Figure 7.8 – *Comparison between execution time SMO-HPS on patches extracted by SMO-HPS and on patches extracted by a sliding window method. The time of the patch extraction process is included.*

7.5 Conclusion

This chapter presents SMO-HPS, a new method of extracting relevant patches from histopathological images and cluster them. It combines subspace clustering and multi-objective techniques to overcome the problems of non-relevant patches. Our approach aims at selecting the relevant patches for classification instead of using the whole image or all selected patches by the sliding window method. We presented two versions of the method, one that deals with numerical datasets, and we tested its performance on synthetic datasets. The second version deal with RGB datasets that we adapted to be compatible with the clustering. This version allows us to extract the relevant patches while grouping them into similar clusters. We also consulted an Anatomo-pathologists to give significance to the group of patches that the method found. The experiments show our method's ability to extract relevant patches while improving the clustering quality and reducing the computational time. To validate the efficiency of the extracted patches by SMO-HPS, we trained the InceptionV3 and VGG16 networks on these patches. The presented results show that the extracted patches by SMO-HPS represent useful histopathological source datasets for training and testing. This method presented also interesting results when combined with the sliding window technique.

Conclusion and Perspective 8

Throughout this Ph.D. thesis, we have tackled fundamental problems in the field of unsupervised machine learning. The first part of the thesis was devoted to presenting state-of-the-art clustering classical data and data stream clustering. We presented different techniques and methods and also detailed comparisons between these methods.

The second part is composed of three chapters:

In the first chapter, we presented S2G-Stream, an efficient method for subspace clustering of an evolving data stream in an online manner with two models of feature and block weighting (global and local). The subspace clustering method with the Global Weighting Model was used as a dimensionality reduction method. This method gives a better clustering of the data stream and provides the features and subspaces contributing to the clustering. Several experiments were conducted to prove the impact on the order of data points and the windows' overlapping to the clustering quality. Experimental evaluation demonstrates the effectiveness and efficiency of S2G-Stream in discovering clusters of arbitrary shapes and relevant features and blocks. The global model weighting gave us a comprehensive view of the features and blocks used to select the best subset of features and enhance the clustering performances.

In the second chapter, we presented MOC-Stream and IMOC-Stream. The first method is a new method for clustering data streams based on a multi-objective algorithm. The latter is an improvement over MOC-Stream to enhance memory and time computation. Unlike those single-objective clustering techniques that have employed only one objective function, IMOC-Stream utilizes two objective functions to find clusters of arbitrary shaped clusters and enhance the clustering quality. IMOC-Stream uses a two-phase process: 1) online phase: creating several clustering solutions based on different algorithms and genetic operators 2) offline phase: construction of an optimal partition from the discovered clusters. We applied our method on large stream datasets and compared it to a different stream clustering algorithm. The experiments show the effectiveness of IMOC-Stream for detecting arbitrary shaped, compact, and well-separated clusters with better execution time.

In the last chapter, we proposed SMO-HPS, a new method of extracting relevant patches from histopathological images and cluster them. It

combines subspace clustering and multi-objective techniques to overcome the problems of non-relevant patches. Our approach aims at selecting the relevant patches for classification instead of using the whole image or all selected patches by the sliding window method. We presented two versions of the technique, one that deals with numerical datasets, and we tested its performance on synthetic datasets. The second version deal with RGB datasets that we adapted to be compatible with the clustering. This version allows us to extract the relevant patches while grouping them into similar clusters. We also consulted Anatomo-pathologists to give significance to the group of patches that the method found. The experiments show our method's ability to extract relevant patches while improving the clustering quality and reducing the computational time.

To validate the efficiency of the extracted patches by SMO-HPS, we trained the InceptionV3 and VGG16 networks on these patches. The presented results show that the extracted patches by SMO-HPS represent useful histopathological source datasets for training and testing. This method also presented interesting results when combined with the sliding window technique.

8.1 Future work

We were interested in the Quantum clustering techniques during this thesis. QC attempts to provide a different solution for clustering problems in data analysis by applying the time-based Schrodinger Equation to study the change of the original data set and the structure of the quantum potential energy function dynamically. The basic idea is to learn the distribution law of sample data in the scale space by studying particles' distribution law in the energy field.

In Quantum Clustering(QC), clustering algorithms are developed based on quantum theory. This is usually done by adapting classical algorithms or their expensive functions to run in quantum computers. The main idea of QC is to study the distribution law of sample data in the scale space based on the distribution law of particles in the energy field. The distribution of particles in the energy fields is determined through their potential energy in the Schrödinger equation. The critical point to connect quantum mechanics and clustering is to assume that each local minimum of the potential energy is the cluster center. Since particles are more likely to appear in the regions with lower potential energy, it is similar to how the same cluster's data points tend to gather in a particular region.

The time-independent Schrödinger equation can be written as in 8.1

$$\left[-\frac{\sigma^2}{2}\nabla^2 + V(r)\right]\Psi(r) = E\Psi(r) \tag{8.1}$$

the parameters of the Schrödinger equation are described in 8.1

Table 8.1 – *Parameters of the Shrödinger equation*

Parameter	Description
σ	clustering kernel parameter
Ψ	stationary wave function
r	position in space of the particle
E	Energy of the particle
V	Potential function

QC algorithm in Horn and Gottlieb [2001] replaced the wave function by the Gaussian kernel-based sum, it starts with assigning a Gaussian Ψ_i with width σ to each data point:

$$\Psi(x_i) = \sum_{j=1}^{N} e^{-\frac{(x_i - x_j)^2}{2\sigma^2}} \tag{8.2}$$

the sum of the individual Gaussian function forms the Parzen window estimator Schløler and Hartmann [1992]:

$$\Psi = \sum_{i} \Psi(x_i) \tag{8.3}$$

Parzen window density estimation is another name for kernel density estimation. It is a non-parametric method for estimating continuous density function from the data.
Then, the potential function that can be used in the Schrodinger equation can be :

$$V = \frac{\sigma^2}{2} \frac{\mathrm{D}^2\Psi(r)}{\Psi(r)} + E \tag{8.4}$$

Where :

$$E = -min \frac{\sigma^2}{2} \frac{\mathrm{D}^2\Psi(r)}{\Psi(r)} \tag{8.5}$$

The use of the parzen window density estimation for each data point is costly in time and memory. We started some experiments to optimize the time and memory computation by replacing the parzen estimator. These researches are still in progress, and we plan on continuing them after the Ph.D. We also intend to apply quantum techniques to clustering data streams as the clustering quality can be improved.

Other challenges in the clustering data stream are discussed in the following Ghesmoune et al. [2016b]:

- **Finding k:** Finding the number of clusters is still an open problem, especially for partitioning-based algorithms. There exist some current methods for this purpose. However, none of them is widely accepted. Some ways generate several partitions of the data and choose the partition that gives the best quality, which automatically determines k.

- **Parameter Requirements:** Current data stream clustering algorithms require parameters such as the number of clusters, some thresholds,

decay factor, window or batch length, etc. Such parameters are susceptible to the input data, and they directly affect the clustering quality. It is a challenge to automatically specify these parameters without domain knowledge, manage them for each cluster separately, and update them according to the data characteristics.

- **Experimental Comparison Environment**: No existing platform can compare two or more different data streams by running them together and evaluating their performances.
- **Different Data Types**: Since the data stream can come from different sources, it can be from other data types. Handling different data types and missing values is a challenging task in data stream clustering. Most of the stream clustering algorithms work with quantitative features and define the similarity based on euclidean distance. Some other approaches convert categorical data into numerical and treat them as quantitative data.
- **Protecting privacy and confidentiality**: Data streams present new challenges and opportunities concerning protecting privacy and confidentiality in data mining. The main objective is to develop data mining techniques that would not uncover information or patterns which compromise confidentiality and privacy obligations. Privacy-by-design seems to be a promising paradigm to use.
- **Handling incomplete information**: The problem of missing values, which corresponds to the incompleteness of features, has been discussed extensively for the offline, static settings. However, only a few works address data streams and especially evolving data streams.
- **Distributed streams**: Data streams are distributed by nature. For learning from distributed data, we need efficient methods in minimizing the communication overheads between nodes. Most importantly, in applications like monitoring, centralized solutions introduce delays in event detection and reaction that can make mining systems inefficient. Many data clustering techniques are not trivial to parallelize. More research is needed with practical and theoretical analysis to provide new methods to develop distributed versions of some techniques.

Contributions

- Mohammed Oualid Attaoui, Hanene Azzag, Mustapha Lebbah, Nabil Keskes ; A New Subspace Multi-Objective Approach for the Clustering and Selection of Regions of Interests in Histopathological Images, IEEE-CEC, Krakow, Poland 2021.
- Nassima Dif, Mohammed Oualid Attaoui, Mustapha Lebbah, Hanene Azzag, Transfer Learning from Synthetic Labels for Histopathological Images Classification, Applied Intelligence 2021.
- Mohammed Oualid Attaoui, Hanene Azzag, Mustapha Lebbah, Nabil Keskes ; Regions of Interest Selection in Histopathological Images using Subspace and Multi-Objective Stream Clustering, Evolutionary Intelligence (Accepted with revisions).
- Mohammed Oualid Attaoui, Hanene Azzag, Mustapha Lebbah, Nabil Keskes: Subspace data stream clustering with global and local weighting models. Neural Comput. Appl. 33(8): 3691-3712 (2021).
- Mohammed Oualid Attaoui, Hanene Azzag, Mustapha Lebbah, Nabil Keskes: Multi-objective data stream clustering. GECCO Companion 2020: 113-114
- Mohammed Oualid Attaoui, Mustapha Lebbah, Nabil Keskes, Hanene Azzag, Mohammed Ghesmoune: Soft Subspace Growing Neural Gas pour le Clustering de Flux de Données. EGC 2020: 441-448
- Mohammed Oualid Attaoui , Mustapha Lebbah, Nabil Keskes, Hanene Azzag, Mohammed Ghesmoune: Soft Subspace Growing Neural Gas for Data Stream Clustering. ICANN (4) 2019: 569-580
- Mohammed Oualid Attaoui, Mustapha Lebbah, Nabil Keskes, Hanene Azzag, Mohammed Ghesmoune: Soft Subspace Topological Clustering over Evolving Data Stream. WSOM+ 2019: 225-230

Appendices

Open Source Clustering Libraries

A

Clustering4Ever

Clustering4Ever[1]. is a free open source Scala/Spark library for clustering. It is a fast and easy to use and test library that can be integrated via an API. It offers several notebooks to help test some algorithms and datasets. The API not only presents a large variety of clustering algorithms but also gives the possibility to use these algorithms with different types of data (continuous, binary, and mixed data). The library also implements Quality measures and different distances measures.

Scikit-learn

Scikit-learn[2] is a simple and efficient tool for predictive data analysis implemented in Python. The library contains a large variety of clustering algorithms (K-Means, DBSCAN, OPTICS, etc.). Many quality measures (Rand, Accuracy) and preprocessing techniques were implemented.

MLlib-Spark

Apache Spark[3] is a popular open-source platform for large-scale data processing that is well-suited for iterative machine learning tasks. MLlib [Meng et al., 2016] is Spark's open-source distributed machine learning library. MLlib provides efficient functionality for a wide range of learning settings and includes several underlying statistical, optimization, and linear algebra primitives. Its an efficient and easy-to-use library that can be integrated into many programming languages like R, Scala, Python, .etc. It includes a clustering library with K-means, GMM, etc.

Weka

Weka [Hall et al., 2009] is an open-source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It supports several clustering algorithms such as

[1] https://github.com/Clustering4Ever/Clustering4Ever
[2] https://scikit-learn.org/
[3] https://spark.apache.org/

EM, Filtered Clusterer, Hierarchical Clusterer, Simple KMeans, and so on. Many feature selection techniques and quality measures are implemented as well.

ELKI

ELKI [Schubert and Zimek, 2019] is an open-source (AGPLv3) data mining software written in Java. The focus of ELKI is research in algorithms, emphasizing unsupervised methods in cluster analysis, and outlier detection. To achieve high performance and scalability, ELKI offers data index structures such as the R*-tree that can provide major performance gains. ELKI is designed to be easy to extend for researchers and students in this domain and welcomes contributions of additional methods. ELKI aims to provide a large collection of highly parameterizable algorithms to allow easy and fair evaluation and benchmarking of algorithms.

Cluster

Cluster[4] is an R package. It includes many clustering algorithms (PAM, CLARA,..), Quality Measures (Silhouette,..), Similarity and Dissimilarity measures, etc.

[4]https://cran.r-project.org/web/packages/cluster/cluster.pdf

Stream Datasets

B

Stream clustering was applied in various fields such as financial transactions, telephone records, sensor network monitoring, telecommunications, website analysis, weather monitoring, and e-business. [Silva et al., 2013] proposed several real-time datasets for stream mining. We discuss them in the following:

KDD'99 was used for The Third International Knowledge Discovery and Data Mining Tools Competition. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment [1]. This dataset represents good test to evaluate stream clustering methods due to its large size.

Sensor networks are spatially distributed separate sensors to monitor physical or environmental conditions, like temperature, sound, pressure, etc., as well as to cooperatively push their data through the network to a base station. The WSN is built of nodes from a few to several hundred or even thousand, where each node is connected to each other sensors [Mamalis et al., 2009]. Data is collected from a set of sensors distributed all around the network. Sensors can send information at different time scales, speed, and granularity. Data continuously flow eventually at high speed, in a dynamic and time-changing environment. These characteristics made sensor networks data one of the most used datasets for stream mining.

Text mining is popularly used in knowledge-driven organizations. It is the process of examining extensive collections of texts to discover new information. Clustering micro-blogging text streams (e.g., Twitter) to obtain temporal and geo-spatial features of real-world events is the most used approach in stream text mining. The extracted characteristics can be sentiments, opinions, etc. [Aggarwal and Yu, 2006] constructed a stream from a number of documents obtained from a 1996 scan of the Yahoo! taxonomy. Considering that Web pages at a given node in the hierarchy are crawled at once, the Web pages are also contiguous by their particular class, as defined by the Yahoo! labels. Clustering text data streams is useful with many applications, such as newsgroup filtering, text crawling, document organization, and topic detection [Silva et al., 2013].

CoverType[2] dataset is used to predict forest cover type from cartographic variables only. This dataset contains a total of 581,012 observations

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://archive.ics.uci.edu/ml/datasets/covertype

with 54 attributes (10 quantitative, four binary values for wilderness areas, and 40 binary soil types). Each observation is labeled as one of seven forest cover types.

Synthetic datasets showed effectiveness in testing hypotheses like the ability to detect arbitrary shaped clusters, noise robustness, and scaling for high dimensionality. This kind of data can be generated in many ways like varying Gaussian distributions [Wan et al., 2008]; using IBM synthetic data generator [Ong et al., 2004], or other dataset formed by arbitrarily shaped clusters.

Table B.1 – *Datasets used in stream clustering*

Datasets	References
KDD'Cup 99	[Chen and Tu, 2007] [Aggarwal and Philip, 2008] [Zhu et al., 2010]
Sensor Networks	[Da Silva et al., 2012] [Gama et al., 2011]
Text Datasets	[Aggarwal and Yu, 2006] [Liu et al., 2008]
CoverType	[Aggarwal et al., 2004] [Kranen et al., 2009] [Carnein et al., 2017]
Synthetic Datasets	[Ong et al., 2004] [Wan et al., 2008] [Al Aghbari et al., 2012]

Stream processing frameworks

C

Recently, many stream processing frameworks have been presented. These frameworks allow us to develop a whole stream application. These frameworks are classified into two main categories: The first classification, called real-time streaming data processing frameworks, incorporates Apache S4[1], Apache Storm[2], Apache Samza[3], and Apache Flink[4]. Such frameworks process the streaming data on a tuple-by-tuple premise in which each tuple is handled as it arrives. Conversely, the frameworks from the second class, for example, Spark Streaming, gather data in certain time intervals and process them in batches. These frameworks are called micro-batch streaming data processing systems [Alshaer, 2019]. We present in the following the frameworks mentioned above.

Real-Time Streaming Data Processing Frameworks

Apache S4

S4 (Simple Scalable Streaming System) is a general-purpose, distributed, scalable, partially fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous, unbounded streams of data. It provides a simple Programming Interface for processing data streams. The goals of this platform [Neumeyer et al., 2010] is to:

- Design a cluster with high availability that can scale using commodity hardware.
- Minimize latency by using local memory in each processing node and avoiding disk I/O bottlenecks.
- Use a decentralized and symmetric architecture
- Use a pluggable architecture to keep the design as generic and customizable as possible.
- Make the design science-friendly, that is, easy to program and flexible.

[1] http://incubator.apache.org/s4/
[2] http://storm.incubator.apache.org/
[3] http://samza.apache.org/
[4] https://flink.apache.org/.

Apache Storm

Apache Storm is a scalable free and open-source distributed system that aims at providing a framework for real-time stream processing, which additionally accomplishes adaptability and adaptation to internal failure. Similar to Hadoop, Storm can be deployed on a cluster of heterogeneous machines and can be used with any programming language. This tool offers specific new business opportunities, including real-time customer service management, data monetization, operational dashboards, or cybersecurity analysis, and threat detection.

Storm has five important features that enable it to support real-time data processing:

- Speed - one million 100-byte messages processed every second per node.
- Scalability - a parallel computing base that runs on a cluster of machines
- Fault tolerance - when workers break down, Storm automatically restarts them. If one node is unavailable, the worker will be restarted on another node.
- Reliability - Storm guarantees that each tuple, or data point, will be processed at least once, or only once. Messages are replayed only if they fail.
- Easy to use - Standard configurations allow production from day one. Once deployed, Storm is simple to use.

Apache Samza

Apache Samza is a distributed stream processing framework developed by LinkedIn[5] in 2013. It provides the following features:

- High performance: Samza provides extremely low latencies and high throughput to analyze data instantly.
- Horizontally scalable: Scales to several terabytes of state.
- Easy to operate: Samza is easy to operate with flexible deployment options - YARN, Kubernetes or standalone.
- Powerful APIs: Rich APIs to build applications.
- Write once, Run everywhere : Ability to run the same code to process both batch and streaming data.
- Pluggable architecture: Integrates with several sources including Kafka, HDFS, AWS.

[5]www.LinkedIn.com

Apache Flink

Flink is a high-performance, scalable batch and stream processing engine which can process a really large amount of data with ease. Flink streaming applications are programmed via a DataStream API that uses Java or Scala. These languages, as well as Python, also allow programming on an additional DataSet API for processing static data. Flink can be deployed in stand-alone mode on a single Java Virtual Machine (JVM), in YARN-based Hadoop clusters, or on Cloud-based systems.

Flink's core runtime environment supports a pipelined, flow-based architecture. It also includes an iterative data processing method for machine learning and various analytical applications. Dedicated libraries and APIs are provided for the development of machine learning programs, as well as for a variety of uses, including string management and graphical element processing. Another API focuses on the integration of Hadoop applications.

Micro-Batch Streaming Data Processing Frameworks

Spark Streaming

Spark Streaming[6] [Zaharia et al., 2012] is a spark module that processes data stream as batches through a similar functional interface to Spark, such as map, filter, reduce, etc. Spark Streaming runs streaming computations as a series of short batch jobs on RDDs within a programming model called discretized streams (D-Streams).

With Spark Streaming, a context is initialized with a duration. The framework will accumulate data during this duration and then produce a small RDD (Resilient Distributed Dataset). This accumulation / RDD production cycle will be repeated until the program is stopped. This is called micro-batches, as opposed to processing events one by one.

Spark Streaming is, therefore, here opposed to Apache Storm: Storm offers real-time processing of events while Spark Streaming will add a delay between the arrival of a message and its processing.

However, this difference in treatment allows Spark Streaming to offer a guarantee of processing messages in exactly once in normal conditions (each message is delivered once and only once to the program, without loss of messages), and at least once in degraded conditions (a message can be delivered several times, but always without loss). Storm allows to set the guarantee level but, to optimize performance, the at most once mode (each message is delivered at most once, but losses are possible) must be used.

Stream MOA

StreamMOA [Bifet et al., 2010] is an open-source Java-based framework for data stream mining. It contains state-of-the-art algorithms and measures for both data stream classification and clustering. It also embodies

[6] http://spark.apache.org/streaming/

several evaluation criteria and visualization tools. The goal of MOA is a benchmark framework for running experiments in the data stream mining context by proving storable settings for data streams (real and synthetic).

Data stream repositories

D

1. **UCI Knowledge Discovery in Databases Archive**[1] is is an online repository of large datasets containing a large number of datasets with different sizes, types, and analysis fields.

2. **COVID-19 Twitter dataset** The COVID-19 Twitter dataset [Banda et al., 2020][2] consists of 43M+(43.845.712 tweets) collected between March 22^{nd} and March 30^{th}.

3. **KDD Cup Center**[3] is an annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining.

4. **UCR Time-Series Datasets**[4] are maintained by Eamonn Keogh, University California at Riverside.

5. **Meetup (2002)** is a website that allows its users to schedule a meeting. Meetup created a mechanism of invitation, and the responses are taken as a stream. This stream is publicly published and can be used for data stream clustering.

6. **Real World Data in Real Time API**[5]

7. **Twitter Data**[6]

8. **National Weather Service (NWS)** (1870) creates public alerts, watches, warnings, advisories, and similar alternative products within the Common Alerting Protocol (CAP) and Atom Syndication Format (ATOM) (NWS Public Alerts, n.d.). This data can be used as a stream for stream clustering methods.

9. **AirNow** Air Quality Observations[7]

[1] http://kdd.ics.uci.edu/
[2] https://github.com/thepanacealab/covid19_twitter
[3] http://www.sigkdd.org/kddcup/
[4] https://www.cs.ucr.edu/
[5] https://www.hooksdata.io/
[6] https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data
[7] https://docs.airnowapi.org/

Bibliography

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., and Sohler, C. (2012). Streamkm++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–4.

Aggarwal, C. (2009). A framework for clustering massive-domain data streams. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 102–113. IEEE.

Aggarwal, C. C. (2007). *Data streams: models and algorithms*, volume 31. Springer Science & Business Media.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment.

Aggarwal, C. C. and Philip, S. Y. (2008). A framework for clustering uncertain data streams. In *2008 IEEE 24th International Conference on Data Engineering*, pages 150–159. IEEE.

Aggarwal, C. C., Philip, S. Y., Han, J., and Wang, J. (2003). -a framework for clustering evolving data streams. In *Proceedings 2003 VLDB Conference*, pages 81–92. Elsevier.

Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., and Park, J. S. (1999). Fast algorithms for projected clustering. In *ACM SIGMoD Record*, volume 28, pages 61–72. ACM.

Aggarwal, C. C. and Yu, P. S. (2006). A framework for clustering massive text and categorical data streams. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 479–483. SIAM.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998a). *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998b). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. page 12.

Ahmed, R., Dalkılıç, G., and Erten, Y. (2020). Dgstream: High quality and efficiency stream clustering algorithm. *Expert Systems with Applications*, 141:112947.

Al Aghbari, Z., Kamel, I., and Awad, T. (2012). On clustering large number of data streams. *Intelligent Data Analysis*, 16(1):69–91.

Alshaer, M. (2019). *An Efficient Framework for Processing and Analyzing Unstructured Text to Discover Delivery Delay and Optimization of Route Planning in Realtime*. PhD thesis.

Amini, A., Saboohi, H., Ying Wah, T., and Herawan, T. (2014). A fast density-based clustering algorithm for real-time internet of things stream. *The Scientific World Journal*, 2014.

Amini, A. and Wah, T. Y. (2010). Density micro-clustering algorithms on data streams: A review. In *World Congress on Engineering 2012. July 4-6, 2012. London, UK.*, volume 2188, pages 410–414. International Association of Engineers.

Anceaume, E. and Busnel, Y. (2013). Sketch*-metric: Comparing data streams via sketching. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 25–32. IEEE.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

Aresta, G., Araújo, T., Kwok, S., Chennamsetty, S. S., Safwan, M., Alex, V., Marami, B., Prastawa, M., Chan, M., Donovan, M., et al. (2019). Bach: Grand challenge on breast cancer histology images. *Medical image analysis*, 56:122–139.

Attaoui, M. O., Azzag, H., Lebbah, M., and Keskes, N. (2020). Subspace data stream clustering with global and local weighting models. *Neural Computing and Applications*, pages 1–22.

Attaoui, M. O., Lebbah, M., Keskes, N., Azzag, H., and Ghesmoune, M. (2019a). Soft subspace growing neural gas for data stream clustering. In *International Conference on Artificial Neural Networks*, pages 569–580. Springer.

Attaoui, M. O., Lebbah, M., Keskes, N., Azzag, H., and Ghesmoune, M. (2019b). Soft subspace topological clustering over evolving data stream. In *International Workshop on Self-Organizing Maps*, pages 225–230. Springer.

Azzag, H., Monmarche, N., Slimane, M., and Venturini, G. (2003). Anttree: A new model for clustering with artificial ants. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 4, pages 2642–2647. IEEE.

Bahri, M. (2020). *Improving IoT data stream analytics using summarization techniques*. PhD thesis, Institut polytechnique de Paris.

Banda, J. M., Tekumalla, R., Wang, G., Yu, J., Liu, T., Ding, Y., and Chowell, G. (2020). A large-scale covid-19 twitter chatter dataset for open scientific research–an international collaboration. *arXiv preprint arXiv:2004.03688*.

Benhammou, Y., Tabik, S., Achchab, B., and Herrera, F. (2018). A first study exploring the performance of the state-of-the art cnn model in the problem of breast cancer. In *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, pages 1–6.

Bhatnagar, V., Kaur, S., and Chakravarthy, S. (2014). Clustering data streams using grid-based synopsis. *Knowledge and information systems*, 41(1):127–152.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2010). Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the First Workshop on Applications of Pattern Analysis*, pages 44–50.

Billot, R., Suchier, H.-M., and Lallich, S. (2008). Une approche ensembliste inspirée du boosting en classification non supervisée. In *EGC*, pages 361–372.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Bohm, C., Railing, K., Kriegel, H.-P., and Kroger, P. (2004). Density connected clustering with local subspace preferences. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 27–34. IEEE.

Borutta, F., Kazempour, D., Mathy, F., Kröger, P., and Seidl, T. (2020). Detecting arbitrarily oriented subspace clusters in data streams using hough transform. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 356–368. Springer.

Bouguelia, M.-R., Belaïd, Y., and Belaïd, A. (2013). An adaptive incremental clustering method based on the growing neural gas algorithm. In *2nd International Conference on Pattern Recognition Applications and Methods - ICPRAM*, pages 42–49.

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Carnein, M., Assenmacher, D., and Trautmann, H. (2017). An empirical comparison of stream clustering algorithms. In *Proceedings of the computing frontiers conference*, pages 361–366.

Carnein, M. and Trautmann, H. (2018). evostream–evolutionary stream clustering utilizing idle times. *Big data research*, 14:101–111.

Chan, E. Y., Ching, W. K., Ng, M. K., and Huang, J. Z. (2004). An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37(5):943–952.

Chen, H., Shi, B., Qian, J., and Chen, Y. (2010). Wavelet synopsis based clustering of parallel data streams. *J Softw*, 21(4):644–658.

Chen, J., Chen, P., and Sheng, X. G. (2013). A sketch-based clustering algorithm for uncertain data streams. *Journal of Networks*, 8(7):1536–1542.

Chen, L., Duan, H., Fan, Y., and Wei, C. (2020). Multi-objective clustering analysis via combinatorial pigeon inspired optimization. *Science China Technological Sciences*, pages 1–12.

Chen, X., Ye, Y., Xu, X., and Huang, J. Z. (2012). A feature group weighting method for subspace clustering of high-dimensional data. *Pattern Recognition*, 45(1):434–446.

Chen, Y. and Tu, L. (2007). Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

Da Silva, A., Chiky, R., and Hébrail, G. (2012). A clustering approach for sampling data streams in sensor networks. *Knowledge and Information Systems*, 32(1):1–23.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.

de Andrade Silva, J. and Hruschka, E. R. (2011). Extending k-means-based algorithms for evolving data streams with variable number of clusters. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 2, pages 14–19. IEEE.

Deng, Z., Choi, K.-S., Jiang, Y., Wang, J., and Wang, S. (2016). A survey on soft subspace clustering. *Information Sciences*, 348:84–106.

Domeniconi, C. and Al-Razgan, M. (2009). Weighted cluster ensembles: Methods and analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(4):17.

Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters.

Dutta, D., Sil, J., and Dutta, P. (2019). Automatic clustering by multi-objective genetic algorithm with numeric and categorical features. *Expert Systems with Applications*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fathzadeh, R. and Mokhtari, V. (2013). An ensemble learning approach for data stream clustering. In *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*, pages 1–6. IEEE.

Ferreira, C. A., Melo, T., Sousa, P., Meyer, M. I., Shakibapour, E., Costa, P., and Campilho, A. (2018). Classification of breast cancer histology images through transfer learning using a pre-trained inception resnet v2. In *International Conference Image Analysis and Recognition*, pages 763–770. Springer.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172.

Foucault, M. (1966). *Les mots et les choses*, volume 42. Gallimard Paris.

Frank, A. and Asuncion, A. (2010). Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of information and computer science*, 213.

Fränti, P., Virmajoki, O., and Hautamäki, V. (2006). Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11):1875–1881.

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976.

Friedman, J. H. and Meulman, J. J. (2004). Clustering objects on subsets of attributes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):815–849.

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in neural information processing systems*, pages 625–632.

Gama, J., Rodrigues, P. P., and Lopes, L. (2011). Clustering distributed sensor data streams using local processing and reduced communication. *Intelligent Data Analysis*, 15(1):3–28.

Gançarski, P., Blansche, A., and Wania, A. (2008a). Comparison between two coevolutionary feature weighting algorithms in clustering. *Pattern Recognition*, 41(3):983–994.

Gançarski, P., Blansche, A., and Wania, A. (2008b). Comparison between two coevolutionary feature weighting algorithms in clustering. *Pattern Recognition*, 41(3):983–994.

Garofalakis, M., Gehrke, J., and Rastogi, R. (2002). Querying and mining data streams: you only get one look a tutorial. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 635–635.

Garza-Fabre, M., Handl, J., and Knowles, J. (2017). An improved and more scalable evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 22(4):515–535.

Ghesmoune, M., Lebbah, M., and Azzag, H. (2015). Micro-batching growing neural gas for clustering data streams using spark streaming. *Procedia Computer Science*, 53:158–166.

Ghesmoune, M., Lebbah, M., and Azzag, H. (2016a). A new growing neural gas for clustering data streams. *Neural Networks*, 78:36–50.

Ghesmoune, M., Lebbah, M., and Azzag, H. (2016b). State-of-the-art on clustering data streams. *Big Data Analytics*, 1(1):13.

Gong, C., Chen, H., He, W., and Zhang, Z. (2017). Improved multi-objective clustering algorithm using particle swarm optimization. *PloS one*, 12(12):e0188815.

Guha, S., Rastogi, R., and Shim, K. (1998). Cure: an efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84.

Guha, S., Rastogi, R., and Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366.

Hahsler, M. and Bolaños, M. (2016). Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1449–1461.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Hamza, H., Belaïd, Y., Belaïd, A., and Chaudhuri, B. B. (2008). Incremental classification of invoice documents. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE.

Handl, J. and Knowles, J. (2004). Evolutionary multiobjective clustering. In *International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer.

Handl, J. and Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE transactions on Evolutionary Computation*, 11(1):56–76.

Handl, J. and Meyer, B. (2007). Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113.

Hassani, M., Kranen, P., Saini, R., and Seidl, T. (2014). Subspace anytime stream clustering. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, pages 1–4.

Hassani, M. and Seidl, T. (2016). Clustering big data streams: recent challenges and contributions. *it-Information Technology*, 58(4):206–213.

Hassani, M., Spaus, P., Gaber, M. M., and Seidl, T. (2012). Density-based projected clustering of data streams. In *International Conference on Scalable Uncertainty Management*, pages 311–324. Springer.

Horn, D. and Gottlieb, A. (2001). Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Physical review letters*, 88(1):018702.

Huang, J., Ng, M., Hongqiang Rong, and Zichen Li (2005). Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

İnkaya, T., Kayalıgil, S., and Özdemirel, N. E. (2015). Ant colony optimization based clustering methodology. *Applied Soft Computing*, 28:301–311.

Itoh, T., Takei, Y., and Tarui, J. (2003). On the sample size of k-restricted min-wise independent permutations and other k-wise distributions. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 710–719.

Janowczyk, A. and Madabhushi, A. (2016). Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *Journal of pathology informatics*, 7.

Jimenez-del Toro, O., Otálora, S., Andersson, M., Eurén, K., Hedlund, M., Rousson, M., Müller, H., and Atzori, M. (2017). Analysis of histopathology images: From traditional machine learning to deep learning. In *Biomedical Texture Analysis*, pages 281–314. Elsevier.

Kao, Y. and Cheng, K. (2006). An aco-based clustering algorithm. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 340–347. Springer.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Kaymak, S., Helwan, A., and Uzun, D. (2017). Breast cancer image classification using artificial neural networks. *Procedia computer science*, 120:126–131.

Keller, A. and Klawonn, F. (2000). Fuzzy clustering with weighting of data variables. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8(06):735–746.

Kennedy, J. (2006). Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer.

Khan, I., Huang, J. Z., and Ivanov, K. (2016a). Incremental density-based ensemble clustering over evolving data streams. *Neurocomputing*, 191:34–43.

Khan, I., Huang, J. Z., and Ivanov, K. (2016b). Incremental density-based ensemble clustering over evolving data streams. *Neurocomputing*, 191:34–43.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1-3):1–6.

Kranen, P., Assent, I., Baldauf, C., and Seidl, T. (2009). Self-adaptive anytime stream clustering. In *2009 Ninth IEEE International Conference on Data Mining*, pages 249–258. IEEE.

Kranen, P., Assent, I., Baldauf, C., and Seidl, T. (2011). The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, 29(2):249–272.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krleža, D., Vrdoljak, B., and Brčić, M. (2020). Statistical hierarchical clustering algorithm for outlier detection in evolving data streams. *Machine Learning*, pages 1–46.

Krzanowski, W. J. and Lai, Y. (1988). A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, pages 23–34.

Kuo, R. and Zulvia, F. E. (2020). Multi-objective cluster analysis using a gradient evolution algorithm. *Soft Computing*, pages 1–15.

Law, M. H., Topchy, A. P., and Jain, A. K. (2004). Multiobjective data clustering. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE.

Li, R., Wang, C., Liao, F., and Zhu, H. (2020). Rcd+: A partitioning method for data streams based on multiple queries. *IEEE Access*, 8:52517–52527.

Liu, Y.-B., Cai, J.-R., Yin, J., and Fu, A. W.-C. (2008). Clustering text data streams. *Journal of computer science and technology*, 23(1):112–128.

Lorbeer, B., Kosareva, A., Deva, B., Softić, D., Ruppel, P., and Küpper, A. (2016). A-birch: automatic threshold estimation for the birch clustering algorithm. In *INNS conference on Big Data*, pages 169–178. Springer.

Lu, Y., Sun, Y., Xu, G., and Liu, G. (2005). A grid-based clustering algorithm for high-dimensional data streams. In *Advanced Data Mining and Applications*, pages 824–831. Springer.

Lucasius, C. B., Dane, A. D., and Kateman, G. (1993). On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasiblity and comparison. *Analytica Chimica Acta*, 282(3):647–669.

Luo, J., Jiao, L., Shang, R., and Liu, F. (2016). Learning simultaneous adaptive clustering and classification via moea. *Pattern Recognition*, 60:37–50.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of BSMSP*, pages 281–297.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Maguolo, G., Nanni, L., and Ghidoni, S. (2019). Ensemble of convolutional neural networks trained with different activation functions. *arXiv preprint arXiv:1905.02473*.

Mamalis, B., Gavalas, D., Konstantopoulos, C., and Pantziou, G. (2009). Clustering in wireless sensor networks. *RFID and Sensor Networks: Architectures, Protocols, Security and Integrations, Y. Zhang, LT Yang, J. Chen, eds*, pages 324–353.

McKusick, K. and Thompson, K. (1990). Cobweb/3: A portable implementation.

Meesuksabai, W., Kangkachit, T., and Waiyamai, K. (2011). Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In *International Conference on Advanced Data Mining and Applications*, pages 27–40. Springer.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

Mescher, A. L. (2018). *Junqueira's basic histology: text and atlas*. McGraw-Hill Education.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Modha, D. S. and Spangler, W. S. (2003). Feature Weighting in k-Means Clustering. page 21.

Mukhopadhyay, A., Maulik, U., and Bandyopadhyay, S. (2015). A survey of multiobjective evolutionary clustering. *ACM Computing Surveys (CSUR)*, 47(4):1–46.

Nanni, L., Brahnam, S., and Maguolo, G. (2019). Data augmentation for building an ensemble of convolutional neural networks. In *Innovation in Medicine and Healthcare Systems, and Multimedia*, pages 61–69. Springer.

Nanni, L., Lumini, A., Ghidoni, S., and Maguolo, G. (2020). Stochastic selection of activation layers for convolutional neural networks. *Sensors*, 20(6):1626.

Nawaz, W., Ahmed, S., Tahir, A., and Khan, H. A. (2018). Classification of breast cancer histology images using alexnet. In *International conference image analysis and recognition*, pages 869–876. Springer.

Neumeyer, L., Robbins, B., Nair, A., and Kesari, A. (2010). S4: Distributed stream computing platform. In *2010 IEEE International Conference on Data Mining Workshops*, pages 170–177. IEEE.

Ng, R. T. and Han, J. (2002). Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016.

Ntoutsi, I., Zimek, A., Palpanas, T., Kröger, P., and Kriegel, H.-P. (2012a). Density-based projected clustering over high dimensional data streams. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 987–998. SIAM.

Ntoutsi, I., Zimek, A., Palpanas, T., Kröger, P., and Kriegel, H.-P. (2012b). Density-based projected clustering over high dimensional data streams. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 987–998. SIAM.

O'callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*, pages 685–694. IEEE.

Ong, K.-L., Li, W., Ng, W.-K., and Lim, E.-P. (2004). Sclope: An algorithm for clustering data streams of categorical attributes. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 209–218. Springer.

Ouattara, M., Keita, N. N., Badran, F., and Mandin, C. (2013). Soft subpace clustering pour données multiblocs basée sur les cartes topologiques auto-organisées som: 2s-som. In *SFDS 2013*.

Papapetrou, O. and Chen, L. (2011). Xstreamcluster: an efficient algorithm for streaming xml data clustering. In *International Conference on Database Systems for Advanced Applications*, pages 496–510. Springer.

Park, N. H. and Lee, W. S. (2007). Grid-based subspace clustering over data streams. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 801–810.

Parsons, L., Haque, E., and Liu, H. (2004). Subspace Clustering for High Dimensional Data: A Review. page 16.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734.

Prudent, Y. and Ennaji, A. (2005). An incremental growing neural gas learns topologies. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 1211–1216. IEEE.

Puschmann, D., Barnaghi, P., and Tafazolli, R. (2016). Adaptive clustering for dynamic iot data streams. *IEEE Internet of Things Journal*, 4(1):64–74.

Raftery, A. E. (1986). A note on bayes factors for log-linear contingency table models with vague prior information. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(2):249–250.

Ren, J., Cai, B., and Hu, C. (2011). Clustering over data streams based on grid density and index tree. *Journal of Convergence Information Technology*, 6:83–93.

Ren, J. and Ma, R. (2009a). Density-based data streams clustering over sliding windows. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 248–252. IEEE.

Ren, J. and Ma, R. (2009b). Density-based data streams clustering over sliding windows. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 248–252. IEEE.

Rodrigues, P. P., Gama, J., and Pedroso, J. P. (2006). Odac: Hierarchical clustering of time series data streams. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 499–503. SIAM.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Sabau, A. S. (2016). Stream clustering using probabilistic data structures. *arXiv preprint arXiv:1612.02701*.

Saini, N., Saha, S., Harsh, A., and Bhattacharyya, P. (2019). Sophisticated som based genetic operators in multi-objective clustering framework. *Applied Intelligence*, 49(5):1803–1822.

Scheid, H. and Schwarz, W. (2009). *elements of linear algebra and calculus*. Springer-Verlag.

Schløler, H. and Hartmann, U. (1992). Mapping neural network derived from the parzen window estimator. *Neural Networks*, 5(6):903–909.

Schubert, E. and Zimek, A. (2019). ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg". *CoRR*, abs/1902.03616.

Shamir, L., Orlov, N., Eckley, D. M., Macura, T. J., and Goldberg, I. G. (2008). Iicbu 2008: a proposed benchmark suite for biological image analysis. *Medical & biological engineering & computing*, 46(9):943–947.

Sheikholeslami, G., Chatterjee, S., and Zhang, A. (1998). Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB*, volume 98, pages 428–439.

Shukla, M., Kosta, Y., and Jayswal, M. (2017a). A modified approach of optics algorithm for data streams. *Engineering, Technology & Applied Science Research*, 7(2):1478–1481.

Shukla, M., Kosta, Y., and Jayswal, M. (2017b). A modified approach of optics algorithm for data streams. *Engineering, Technology & Applied Science Research*, 7(2):1478–1481.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. d., and Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Spanhol, F. A., Oliveira, L. S., Petitjean, C., and Heutte, L. (2015). A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, 63(7):1455–1462.

Spanhol, F. A., Oliveira, L. S., Petitjean, C., and Heutte, L. (2016). Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)*, pages 2560–2567. IEEE.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

Sui, J., Liu, Z., Liu, L., Jung, A., and Li, X. (2020). Dynamic sparse subspace clustering for evolving high-dimensional data streams. *IEEE Transactions on Cybernetics*.

Sun, J. and Binder, A. (2017). Comparison of deep learning architectures for h&e histopathology images. In *2017 IEEE Conference on Big Data and Analytics (ICBDA)*, pages 43–48. IEEE.

Supardi, N. A., Abdulkadir, S. J., and Aziz, N. (2020). An evolutionary stream clustering technique for outlier detection. In *2020 International Conference on Computational Intelligence (ICCI)*, pages 299–304. IEEE.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Tareq, M. and Sundararajan, E. A. (2020). A new density-based method for clustering data stream using genetic algorithm. 1.

Tareq, M., Sundararajan, E. A., and Mohd, M. (2020). Online clustering of evolving data stream based on adaptive chebychev distance. In *Proc. 281st Int. Conf. IIER*, pages 41–46.

Tu, L. and Chen, Y. (2009). Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):1–27.

Udommanetanakit, K., Rakthanmanon, T., and Waiyamai, K. (2007). E-stream: Evolution-based technique for stream clustering. In *International Conference on Advanced Data Mining and Applications*, pages 605–615. Springer.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

Wan, R., Yan, X., and Su, X. (2008). A weighted fuzzy clustering algorithm for data stream. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 1, pages 360–364. IEEE.

Wang, R., Lai, S., Wu, G., Xing, L., Wang, L., and Ishibuchi, H. (2018). Multi-clustering via evolutionary multi-objective optimization. *Information Sciences*, 450:128–140.

Wang, W., Yang, J., Muntz, R., et al. (1997). Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.

Xu, J., Zhou, C., Lang, B., and Liu, Q. (2017). Deep learning for histopathological image analysis: towards computerized diagnosis on cancers. In *Deep Learning and Convolutional Neural Networks for Medical Image Computing*, pages 73–95. Springer.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

Yang, Y., Liu, Z., Zhang, J.-p., and Yang, J. (2012). Dynamic density-based clustering algorithm over uncertain data streams. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2664–2670. IEEE.

Zaharia, M., Das, T., Li, H., Shenker, S., and Stoica, I. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In *Presented as part of the*.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM.

Zhang, X., Furtlehner, C., and Sebag, M. (2008). Data streaming with affinity propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 628–643. Springer.

Zhi, W., Yueng, H. W. F., Chen, Z., Zandavi, S. M., Lu, Z., and Chung, Y. Y. (2017). Using transfer learning with convolutional neural networks to diagnose breast cancer from histopathological images. In *International Conference on Neural Information Processing*, pages 669–676. Springer.

Zhou, A., Cao, F., Qian, W., and Jin, C. (2008). Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15(2):181–214.

Zhu, H., Wang, Y., and Yu, Z. (2010). Clustering of evolving data stream with multiple adaptive sliding window. In *2010 International Conference on Data Storage and Data Engineering*, pages 95–100. IEEE.

Zhu, X., Yao, J., Zhu, F., and Huang, J. (2017). Wsisa: Making survival prediction from whole slide histopathological images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7234–7242.

Zhu, Y. and Shasha, D. (2002). StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time** Work supported in part by US NSF grants IIS-9988345 and N2010: 0115586. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 358–369. Elsevier.