

# **Data Stream Mining and Analysis: Clustering Evolving Data**

Christian Nordahl



Blekinge Institute of Technology thesis for the degree of doctor of  
philosophy series  
No: 2023:XX

# **Data Stream Mining and Analysis: Clustering Evolving Data**

**Christian Nordahl**

Thesis for the degree of Doctor of Philosophy in  
Computer Science



Faculty of Computer Sciences  
Department of Computer Science  
Blekinge Institute of Technology

SWEDEN

2023XX Christian Nordahl  
Faculty of Computer Sciences  
Department of Computer Science  
Publisher: Blekinge Institute of Technology,  
SE-371 79 Karlskrona, Sweden  
Printed by Lenanders, Kalmar, Sweden 2023XX  
ISBN 123-21332-13423-123  
ISSN 1653-2090

”Sometimes when you’re in a dark place, you think you’ve been buried, but you’ve actually just been planted.“

Christine Caine



## ABSTRACT

Streaming data is becoming more prevalent in our society for every passing day. With growing technologies, such as Internet of Things (IoT), and 5G networks, the number of possible data sources are creeping towards infinity. There is a need to develop algorithms that can handle the massive amount of data that is being generated.

Data stream mining is a sub-domain of data mining, where data is mined to gain an understanding of . When we move to streaming data there are restrictions imposed on the algorithms that can be used. Data streams are possibly endless, data arrives in a rapid pace, data instances can often only be processed once or a few times, and the data is often evolving as the data is generated over time.

In this thesis, we explore data-driven techniques to model and analyze evolving data streams. We aim to evaluate existing as well as develop novel technologies to identify approaches that are suitable for use on an individual user level. We focus on the domain of electricity consumption and we apply both supervised and unsupervised learning techniques to the user/system behavior.

More specifically, we investigate the use of regression and clustering algorithms to mine streaming data for user/system behavior patterns. We also design an algorithm that is capable of modeling user/system behavior in a single evolving data stream, that is easy to use and capitalizes on prior knowledge from the history of the stream. Furthermore, we design an clustering algorithm that takes advantage of multiple data streams, where each stream represent a part of the entire system, to model various aspects of the user/system behavior. Finally, we review the current landscape for evaluation of data stream clustering algorithms and identify aspects that should be considered for the future.



---

# Preface

---

## Included Papers

This thesis consists of seven papers. The author has been the main driver in PAPERS I-III and V-VII, and one of the main drivers for PAPER IV. The formatting of the papers included in this thesis has been changed to conform to a common style, no other changes have been performed.

- |          |  |
|----------|--|
| PAPER I  | Nordahl, C., Persson, M, and Grahn, H.. “Detection of residents’ abnormal behaviour by analysing energy consumption of individual households”. In <i>2017 IEEE International Conference on Data Mining Workshops (ICDMW)</i> , 1st International Workshop on AI for Aging, Rehabilitation, and Ambient Assisted Living (ARIAL), November, 2017, New Orleans, Florida, USA, pp. 729-738.<br>DOI: 10.1109/ICDMW.2017.101 |
| PAPER II | Nordahl, C., Boeva, V., Grahn, H., and Persson-Netz, M. “Profiling of Household Residents’ Electricity Consumption Behavior Using Clustering Analysis”. In <i>International Conference on Computational Science</i> , June, 2019, pp. 779-786, Springer, Faro, Portugal. DOI: 10.1007/978-3-030-22750-0_78   |

PAPER III	<u>Nordahl, C.</u> , Boeva, V., Grahn, H., and Persson-Netz, M. "Monitoring Household Electricity Consumption Behaviour for Mining Changes". In <i>3rd International Workshop on AI for Aging, Rehabilitation and Independent Assisted Living (ARIAL)</i> , International Joint Conference on Artificial Intelligence (IJCAI), August, 2019, Macau, China, August, 2019, Macau, China. diva2:1350711
PAPER IV	<u>Boeva, V.</u> and <u>Nordahl, C.</u> "Modeling Evolving User Behavior via Sequential Clustering". In <i>2nd International Workshop on Knowledge Discovery and User Modelling for Smart Cities (UMCit)</i> , The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), September, 2019, Würzburg, Germany. diva2:1352327
PAPER V	<u>Nordahl, C.</u> , Boeva, V., Grahn, H., and Persson-Netz, M. "EvolveCluster: An Evolutionary Clustering Algorithm for Streaming Data". In: <i>Evolving Systems</i> 13(4), 2022, pp. 603-623. DOI: s12530-021-09408-y
PAPER VI	<u>Nordahl, C.</u> , Boeva, V., and Grahn, H.. "MultiStream EvolveCluster". In proceedings of the 36th Canadian Conference on Artificial Intelligence (CANAI), Montreal, Canada, 2023. DOI: 10.21428/594757db.b22e0e9a <i>Ranked as one of the top 3 papers of the conference.</i>
PAPER VII	<u>Nordahl, C.</u> , Boeva, V., Grahn, H., and Persson-Netz, M. "On Evaluation of Data Stream Clustering Algorithms: A Survey". Submitted to the Data Mining and Knowledge Discovery journal.

Other research contributions that are related to this thesis but not included:

PAPER VIII	<u>Nordahl, C.</u> , Boeva, V., Grahn, H., and Persson-Netz, M. "Organizing, Visualizing and Understanding Households Electricity Consumption Data through Clustering Analysis". In <i>2nd International Workshop on AI for Aging, Rehabilitation and Independent Assisted Living (ARIAL)@ IJCAI'18</i> , July, 2018, Stockholm, Sweden, google.com/view/arial2018/accepted-papersprogram
------------	---

---

## Acknowledgements

---



---

# Contents

---

Abstract . . . . .	i
Preface . . . . .	iii
Acknowledgements . . . . .	v
<b>1 Introduction</b>	<b>1</b>
1.1 Research Questions . . . . .	2
1.2 Outline . . . . .	3
<b>2 Background</b>	<b>5</b>
2.1 Machine Learning . . . . .	5
2.2 Clustering Analysis . . . . .	9
2.3 Data Mining and Knowledge Discovery . . . . .	13
2.4 Data Stream Mining . . . . .	14
2.5 Outlier Detection . . . . .	14
<b>3 Related Work</b>	<b>17</b>
3.1 Electricity Consumption Behaviors . . . . .	17
3.2 Clustering Streaming Data . . . . .	18
3.3 Clustering Multiple Data Streams . . . . .	20
<b>4 Methodology</b>	<b>23</b>
4.1 The Scientific Method and Engineering Design Process . . . . .	23
4.2 Research Methodology . . . . .	24
4.3 Data sets . . . . .	24
4.4 Evaluation Measures . . . . .	26
4.5 Validity Threats . . . . .	29

<b>5 Proposed Data Mining Algorithms and Results</b>	<b>31</b>
5.1 Modeling User/System Behavior . . . . .	31
5.2 Modeling Single Evolving Data Streams . . . . .	33
5.3 Modeling Multiple Evolving Data Streams . . . . .	35
5.4 Evaluating Data Stream Clustering Algorithms . . . . .	37
5.5 Summary . . . . .	38
<b>6 Conclusions and Future Work</b>	<b>41</b>
<b>Bibliography</b>	<b>43</b>
<b>7 Detection of Residents' Abnormal Behaviour by Analysing Energy Consumption of Individual Households</b>	<b>51</b>
<i>Christian Nordahl, Marie Persson, and Håkan Grahn</i>	
7.1 Introduction . . . . .	51
7.2 Background . . . . .	52
7.3 Data Analysis . . . . .	55
7.4 Approach . . . . .	57
7.5 Experiments . . . . .	62
7.6 Results and Analysis . . . . .	64
7.7 Discussions . . . . .	67
7.8 Conclusions and Future Work . . . . .	68
References . . . . .	69
<b>8 Profiling of Household Residents' Electricity Consumption Behavior using Clustering Analysis</b>	<b>73</b>
<i>Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz</i>	
8.1 Introduction . . . . .	73
8.2 Clustering Analysis Approach . . . . .	75
8.3 Experiments and Results . . . . .	76
8.4 Conclusions and Future Work . . . . .	81
References . . . . .	82

<b>9 Monitoring Household Electricity Consumption Behavior for Mining Changes</b>	<b>85</b>
<i>Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz</i>	
9.1 Motivation and State of the Art . . . . .	86
9.2 The Proposed Approach and Methods . . . . .	87
9.3 Initial Evaluation and Results . . . . .	90
9.4 Conclusions and Future Work . . . . .	94
References . . . . .	94
<b>10 Modeling Evolving User Behavior via Sequential Clustering</b>	<b>97</b>
<i>Veselka Boeva and Christian Nordahl</i>	
10.1 Introduction . . . . .	98
10.2 Modeling Evolving User Behavior via Sequential Clustering .	99
10.3 Case Study: Modeling Household Electricity Consumption Behavior . . . . .	102
10.4 Conclusions and Future Work . . . . .	106
References . . . . .	106
<b>11 EvolveCluster: An Evolutionary Clustering Algorithm for Streaming Data</b>	<b>109</b>
<i>Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz</i>	
11.1 Introduction . . . . .	109
11.2 Background . . . . .	111
11.3 Related Work . . . . .	117
11.4 An Evolutionary Clustering Algorithm . . . . .	121
11.5 Data and Experimental Designs . . . . .	125
11.6 Results and Analysis . . . . .	131
11.7 Discussion . . . . .	141
11.8 Conclusions . . . . .	144
References . . . . .	146

<b>12 MultiStream EvolveCluster</b>	<b>153</b>
<i>Christian Nordahl, Veselka Boeva, and Håkan Grahn</i>	
12.1 Introduction . . . . .	153
12.2 Background . . . . .	155
12.3 Related Work . . . . .	157
12.4 MultiStream EvolveCluster . . . . .	158
12.5 Empirical Evaluation . . . . .	161
12.6 Results, Analysis, and Discussion . . . . .	164
12.7 Conclusions and Future Work . . . . .	169
References . . . . .	170
<b>13 On Evaluation of Data Stream Clustering Algorithms: A Survey</b>	<b>173</b>
<i>Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz</i>	
13.1 Introduction . . . . .	173
13.2 Data Stream Clustering . . . . .	175
13.3 Evaluating Clustering Algorithms . . . . .	182
13.4 Evaluation Methodologies . . . . .	187
13.5 Discussion . . . . .	192
13.6 Conclusions . . . . .	198
.1 CVIs and Evaluation Measures Used in the Domain. . . . .	202
.2 Datasets Used in the Domain. . . . .	204
.3 Raw Data Tables from Literature Review. . . . .	206
References . . . . .	206

---

## Introduction

---

Streaming data has been on the rise over the last couple of decades. The size and price of electrical components are constantly decreasing which increases the possibilities of data to collect. For instance, it was not common 2 decades ago for electricity consumption to be measured and stored on a household level, but in Sweden today there is a 99% coverage of smart meters that are starting to perform just that in real time. Generally, the expansion of Internet of Things (IoT) provide us with many data sources that were previously unthinkable.

With the increase of data availability, we also have to manage these new data sources [1, 2]. In data stream mining, we operate under the assumption that each stream is possibly endless, has a rapid arrival rate of data instances, and contains evolving concepts. These hurdles often require specialized algorithms to have low computational costs and adaptability for evolving concepts [3, 4]. However, benefits of the new data sources are immense and we can now use combinations of data streams to gather separate views of a system [5].

In this thesis, we investigate and design data mining techniques and algorithms to model behaviors present in evolving data streams. We focus on both single data stream modeling and multistream modeling.

Being able to evaluate these algorithms is of great importance, both for making sure results are reproducible and to compare algorithms in an easier more consistent way. The types of data that can be within the data clustering domain is endless, every application domain that produces data is subsequently a possible endeavour for research. As we have so many types of applicable domains, we have as many datasets. With the free lunch theorem there is no single technique to rule them all. Thus, this thesis also investigate the common practices of evaluation of clustering algorithms that operate on

streaming data, to identify possible ways forward.

### 1.0.1 Scope and Aim

The main focus of the thesis is to model the evolving concepts that appear over time in streaming data using machine learning techniques. More specifically, to develop and apply clustering algorithms and techniques that can accurately model the behavior present in data streams. The data streams of interest for this thesis are not the most rapid, rather we are interested in those data streams where flexibility in terms of computational efforts and storage requirements is available. Additionally, for those uses where a direct incremental update is not needed, rather an overview of the behavior over a time period.

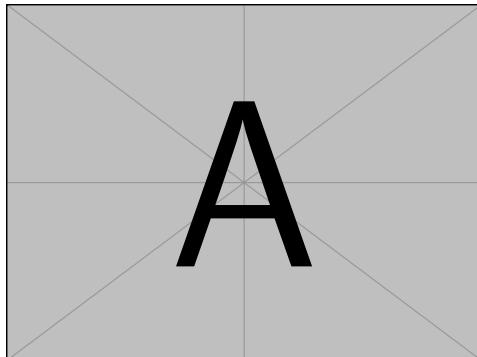


Figure 1.1: *PLACEHOLDER*. Will create figure to visualize the scope and aim of the thesis and how each paper contributes to that.

## 1.1 Research Questions

Given the scope of the thesis, the research questions we aim to answer are:

**RQ 1:** *How can streaming data be modeled for profiling user/system behavior patterns?*

Chapter 7 attempted to address the question with the use of regression models. Several models were evaluated using carefully extracted features chosen by their correlation values, and various data granularities. The regression models were also investigated for use with detection of abnormal behavior or anomalies. Chapter 8 instead addressed the question by applying

clustering models to model the behavior. Here, dissimilarity measures were also investigated for their impact in the modeling.

**RQ 2:** *How can we model and monitor user/system behavior in a single evolving data stream?*

We address this question in Chapters 9 and 10 in which we sequentially cluster data segments. Each segment is clustered with the use of prior knowledge, i.e., cluster centroids from the previous segment. In Chapter 11 it was further addressed by incorporating mechanics to handle the evolving and dynamic nature of data streams, creating a novel algorithm named EvolveCluster. Specific splitting criterion's were introduced to manage new emerging behaviors over time.

**RQ 3:** *How can we take advantage of multiple data streams to model the behavior of a user/system?*

This question is addressed in Chapter 12. Here, an extension of EvolveCluster was proposed, named MultiStream EvolveCluster, that incorporates knowledge from multiple data streams to create a global clustering solution. Each stream provides its own view of the system and is clustered individually, which allow for analysis of each individual part of the system. But, the consensus clustering solution provides an overall view of how the total system is performing and can be analyzed to see how each view relates to one another.

**RQ 4:** *Which cluster evaluation techniques are suitable to be used to evaluate data stream clustering algorithms?*

A survey was conducted to address this question, which is located in Chapter 13. During the earlier studies it was identified that a common practice for evaluation was hard to identify without a thorough investigation of the domain.

## 1.2 Outline

The remainder of this thesis is organized as follows:

## 1. INTRODUCTION

---

**Chapter 2 - Background:** Here we provide the relevant background information that this thesis is built upon.

**Chapter 3 - Related Work:** Related work is presented in this chapter to provide the scenery for this thesis.

**Chapter 4 - Methodology:** This chapter presents the research methodology used in this thesis. We also present the used datasets and evaluation measures. Finally, we discuss the validity threats of the thesis.

**Chapter 5 - Proposed Data Mining Algorithms and Results:** Here we present the summarized results from the seven studies and answer the research questions.

**Chapter 6 - Conclusions and Future Work:** This chapter concludes the thesis with a summary of its findings and possible ways forward for the work presented in the thesis.

## Background

This chapter presents the necessary background information to understand the remainder of the thesis. We start by introducing machine learning and its subsets of domains that are used in this thesis. We continue by formalizing data mining and outlier detection, and we conclude with a thorough explanation of clustering analysis and the additional tools needed to use clustering.

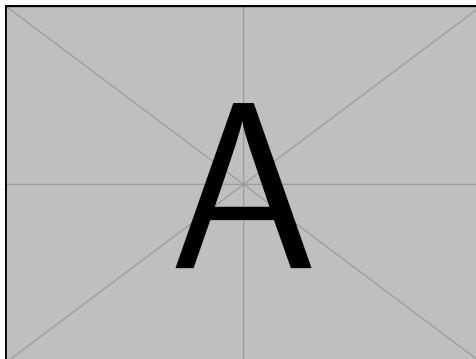


Figure 2.1: *PLACEHOLDER*. Will create figure to visualize how the background helps to understand the remainder of the thesis, and how each area corresponds to each paper/research question..

### 2.1 Machine Learning

Computers were first invented to help humans with performing calculations. Since then they have evolved, and their support and capabilities have made them integral to our daily lives. They are great at tasks where we can define rules and processes in program code, but some tasks are not easy to explicitly define a process for. For instance, the task of object identification in pictures. The simple task of seeing if a cat is present or not in a picture

## 2. BACKGROUND

---

is seemingly easy for humans, but what is it *really* that makes us know that an animal is a cat?

Because we have trouble relaying our process to determine what is a cat and what is not, we cannot instruct a computer to do it. Instead of instructing a computer *how* to identify and differentiate between different objects, we can just provide data and their answers to the computer and let it identify the traits by itself. This is a simple explanation of what Machine learning (ML) is. In the cat image example, we would feed images and specify which images contain cats, letting the computer identify which traits are important for identifying cats [6].

The first mention of ML in the academic literature was back in 1959 by Samuel [7], where principles of ML were presented while he designed a program to learn how to play the game Checkers. Games have traditionally been a common platform for conducting ML research. Rules are often simple, and the results are easily understood, especially compared to real-life scenarios. One of the great achievements of computers in games would be Deep Blue [8], where IBM created a machine that beat the reigning chess world champion, Garry Kasparov. Chess was long seen as hard for computers to overcome human capabilities, as the number of possible legal moves amounts to around  $10^{40}$ , and we tended to use intuition and experience to influence our moves instead of analyzing every possible move. Although Deep Blue was not directly using ML, it started to show the capabilities of the computer were more than just producing spreadsheets. Go, a game that has an enormous number of possible moves, approximated to around  $10^{170}$ , was often deemed impossible for computers to master. However, AlphaGo [9] and its successor AlphaGo Zero [10] managed to beat the 18-time world champion Lee Sodol in 2016. These two algorithms took full advantage of the ML paradigm, where the original AlphaGo taught itself to play by studying millions of previously played games between humans. The successor AlphaGo Zero managed by playing against itself.

A formal definition for ML was provided by Mitchell [11], where learning is defined as a computer program that can “[...] learn from experience  $E$  with regards to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ”. If we refer to our earlier example of identifying if a cat is present in an image, the task  $T$  is to identify whether or not a cat is present in the image.

To train the model, the performance  $P$  can be measured as the ratio of correctly classified images. The experience  $E$  is then the data set of images containing cats and other animals where each image is labeled if it contains a cat or not.

There are three main categories we can place ML algorithms in:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

Depending on what type of data you have available and the intention of the application, each category has its benefits and drawbacks. In this thesis, we have not used reinforcement learning techniques for the ML tasks studied. Therefore we do not consider the theoretical background of this learning problem. For the interested reader, we refer you to sources such as [12].

### 2.1.1 Supervised Learning

Algorithms that require a data set to contain labels to be trained are put into the category of supervised learning algorithms [13]. The term supervised refers to the guidance from the instructor and the data set itself. Each data set instance has input and target values, and the algorithm learns to map data instances to their corresponding target values. Depending on the output of the supervised learning algorithm, we can distinguish the algorithms between classification algorithms and regression algorithms.

Classification refers to mapping an input to its corresponding target class. Revisiting the previously mentioned image example, we wanted to determine if an image contained a cat. This is a typical binary classification problem, there either is a cat, or there is not. Extending the example also to determine if an image contains a dog, a multiclass classification problem arises. The algorithm now has to distinguish if an image contains a cat, dog, or neither animal. Following the notation of Goodfellow et al. [6], a classification algorithm intends to find a function  $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$  that identifies which of the  $k$  classes an input belongs to.

Regression algorithms also learn from a data set with labels. However, the algorithms approximate the input to a real value instead of a class label

as the target. To solve this task, the regression algorithm aims to learn the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .

This thesis uses Linear Regression and Support Vector Regression, which are explained in more detail below.

**Linear Regression** In simple Linear Regression (LR), the goal is to attempt to model the relationship between an independent and a dependent variable. It is a statistical approach where it is assumed the variables have a linear relationship between them [14]. We can formalize LR as follows

$$y = \beta_0 + \beta_1 x + \epsilon, \quad (2.1)$$

where the dependent variable  $y$  is predicted by the independent variable  $x$ . The random variable  $\epsilon$  is a statistical error term, and  $\beta_0$  and  $\beta_1$  are regression coefficients estimated by observed values of  $x$  and  $y$ . The regression coefficients are estimated by minimizing the sum of squared errors, ordinarily using the least squares method.

More independent variables than one could also predict the dependent variable  $y$ . When multiple independent variables are used to predict the dependent variable, each independent variable has its regression coefficient. This model is called Multivariate LR and is formalized as follows

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon. \quad (2.2)$$

**Support Vector Regression** A state-of-the-art machine learning algorithm called Support Vector Machine (SVM) can solve many classification tasks [15]. The SVM classifies instances by identifying a hyperplane that separates two classes with the largest margin to any point within the training set. Extensions to the SVM have been added to solve more diverse problems, including pattern recognition [16], image analysis [17], and specifically regression [18]. The regression extension is called Support Vector Regression (SVR) and operates similarly to the original SVM. However, instead of identifying a hyperplane to separate data points, the SVR produces a tube that encapsulates the data points that are observed. An interesting aspect of the SVR is that it does not necessarily rely on a linear relationship between the dependent and independent variables. Depending on which kernel function is chosen, non-linear relationships can be modeled by the algorithm.

### 2.1.2 Unsupervised Learning

Algorithms in the unsupervised learning category are left for themselves to investigate for traits in the dataset. Little or no guidance is given to them regarding how or what to identify or investigate. These algorithms learn some useful properties about the data or the data distribution and how to distinguish between instances in the dataset [6]. There are many subcategories within the unsupervised learning category, including latent variable models [19] and clustering analysis [20]. This thesis mostly focuses on clustering algorithms, and we provide a thorough explanation of it in the following section.

## 2.2 Clustering Analysis

In this section, we present the different parts required to conduct a clustering analysis approach. First we present clustering approaches, with a focus on partitioning algorithms. At the end of the section we introduce dissimilarity measures.

### 2.2.1 Clustering Algorithms

Clustering algorithms are designed to identify the underlying structure of a data set and use the detected relationships to create coherent groups. Traditionally, there are five main types of approaches to clustering; partitioning, hierarchical, model-based, density-based, and grid-based [21]. In this thesis, due to the data and the intended task, the focus has been put on partitioning algorithms, specifically  $k$ -medoids.

Partitioning algorithms differ from the other types of algorithms in the sense that they require a parameter that defines how many clusters should be produced. This parameter, usually denoted  $k$ , is not easy to identify in advance. A common approach to identify an appropriate  $k$  is to execute the clustering algorithm multiple times with randomly selected cluster centroids and an increasing  $k$  value. The produced clustering solutions are then evaluated by validation measures to determine the appropriate  $k$  value. The initial cluster centroids do, however, not have to be randomly selected; there are approaches that can estimate suitable cluster centroids, such as [22].

$K$ -means is one of the most prominent examples of a partitioning al-

## 2. BACKGROUND

---

gorithm. The algorithm starts with randomly assigning  $k$  initial cluster centroids, unless the centroids are manually specified, where  $k$  is the given parameter defining the number of clusters to produce. All data points are assigned to the closest centroid, forming  $k$  clusters.  $K$ -means then iteratively improves the solution by:

1. Creating a new centroid by calculating the mean values for each of the data objects in the cluster
2. Re-assigning all data objects to the closest centroid

The optimization takes place until no change has been made or the maximum number of iterations has been surpassed.

Adaptations of  $k$ -means have been proposed, that operate similarly to  $k$ -means but use other types of cluster centroids, such as  $k$ -median and  $k$ -medoids.  $K$ -median defines its cluster centroids by calculating the median values of the cluster objects instead of the mean.  $K$ -medoids, on the other hand, choose the most centrally located object as its cluster centroid, i.e., an actual data object in the cluster is chosen as its core.

For our clustering experiments in this thesis, we have chosen  $k$ -medoids. One clear advantage to this approach is that there is no need to recalculate the distances between the centroid and the other profiles in each iteration because they are all previously known. Additionally, since we are using DTW as a distance measure between the profiles the mean profile would not be an accurate signature of the cluster content. As DTW warps the time axis, the consumption peaks may cancel out each other when the mean is calculated, ending up in a distorted consumption signature. If ED was used as a primary distance measure, as it favors profiles that have their consumption peaks located at the exact same times,  $k$ -means and  $k$ -median would have been reasonable choices.

### 2.2.2 Dissimilarity Measures

Some clustering algorithms require measures that accurately quantify the distinction between data instances, usually ones that are centroid-based algorithms. Depending on what type of data is to be clustered, there are different measures proposed. For instance, Levenshtein Distance [23] to

capture dissimilarities between words, Hamming Distance [24] to compare two equal length symbol strings, and the more general purpose Euclidean Distance [25]. In this thesis, we partly verify the eligibility of our approach data sets containing time-series data. Specifically, several household electricity consumption data sets measured in kWh at specific times during the day.

Traditionally, Euclidean Distance (ED) [25] is one of the more commonly used dissimilarity measures for time-series data. It is fast and easily understandable but also well-performing for many use cases. It is formalized as follows

$$\text{ED}(x, y) = \sqrt{\sum_{i=0}^n (x_i - y_i)^2},$$

where  $x$  and  $y$  are individual time-series in an  $n$ -dimensional space and  $x_i$  and  $y_i$  are individual points in  $x$  and  $y$  respectively. In Figure 2.2, we illustrate how ED calculates the distance between two time-series. The dashed lines show how the dissimilarity is calculated between the two time-series, point by point.

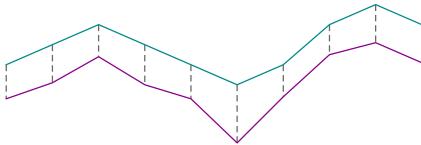


Figure 2.2: Euclidean distance between two time-series.

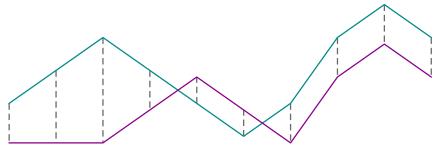


Figure 2.3: Euclidean distance between two time-series that are slightly out of phase.

When the amplitude changes at the same points in time in both time-series, ED identifies a similar pattern. However, as illustrated in Figure 2.3, ED cannot capture patterns that occur slightly out of phase. This is due to the strict nature of the point-wise calculations. The two time-series in Figure 2.3 have similar patterns, but they do not occur at the same times. Elastic dissimilarity measures [26], such as Dynamic Time Warping (DTW) [27] and Longest Common Subsequence (LCSS) [28], address such shortcomings.

DTW builds upon ED, using the same calculation between two points. However, it calculates the distance between all the points of the two time-

series, creating an  $n \times m$  matrix. The two time-series  $x = \{x_1, \dots, x_n\}$  and  $y = \{y_1, \dots, y_m\}$  are aligned in the matrix' axis, one on the bottom and one on the left side. Each matrix cell is populated with the score measuring the distance between the corresponding points in both time-series. The matrix is then navigated through to minimize the total cost, or dissimilarity, identifying the best path through the matrix. The best path  $P = (1, 1), \dots, (i_s, j_s), \dots, (n, m)$  is the path that minimizes the distance between  $x$  and  $y$ , giving the best alignment between the two time-series. Letting DTW align the time-series against each other allows for the identification of similar patterns in the time-series even though they are out of phase from each other. It can be seen as DTW stretches, or warps, the time axis of the time-series data to make them more similar.

The DTW distance between  $q$  and  $p$  can be defined as

$$dtw(x, y) = \frac{1}{n + m} \min_P \left( \sum_{s=1}^k d(i_s, j_s) \right). \quad (2.3)$$

Figure 2.4 shows how DTW matches against several points, where the dotted lines are the calculated shortest distances between the two time-series. The

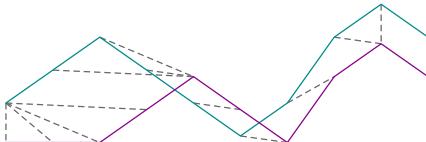


Figure 2.4: Dynamic Time Warping between two series.



Figure 2.5: Aligned Dynamic Time Warping between two time-series. Purple is aligned and matched to green.

warping between the two time-series is visualized in Figure 2.5. The green line is in its original state, and the purple has been warped to its closest match.

The complexity of DTW is, however, exhaustive compared to ED. ED calculates the dissimilarity between  $n$  data points, giving a  $\mathcal{O}(n)$  time complexity. DTW, on the other hand, has a time complexity of  $\mathcal{O}(n \cdot m)$ . Every data point in a time-series can match against many data points in the other, requiring calculation of dissimilarity between all data points.

Canberra distance was designed by Lance et al. [29] and examines the sum of fraction differences between two vectors. We define Canberra distance as

$$d_{canberra}(x, y) = \sum_{i=1}^n \frac{|x_i - y_i|}{|x_i| + |y_i|}, \quad (2.4)$$

where  $x$  and  $y$  are individual vectors in an  $n$ -dimensional space, and  $x_i$  and  $y_i$  are individual points in  $x$  and  $y$  respectively.

The Minkowski distance is a generalization that encompasses ED, Hamming distance, Manhattan distance and Chebyshev distance. It can be defined as

$$d_{minkowski}(x, y) = \sum_{i=1}^n (|x_i - y_i|^p)^{\frac{1}{p}}, \quad (2.5)$$

where  $x$  and  $y$  are individual vectors in an  $n$ -dimensional space, and  $x_i$  and  $y_i$  are individual points in  $x$  and  $y$  respectively.

## 2.3 Data Mining and Knowledge Discovery

Data mining is an exploratory analysis of, often large, data sets to identify relationships and patterns that are useful [30]. It is an iterative process where each finding about the data set gives insight of the data. In this domain, ML is often used as a tool to learn and model the data in different ways. Within data mining, we can define two primary goals, prediction and description [31].

Prediction refers to the ability to predict unknown or future values by using variables in the data set. A model is built by extracting information from the given data set that describes the system at hand. The produced model can then be used to perform predictions or classifications of future states for the system.

On the descriptive side of data mining, the focus turns to finding patterns that describe the data and produce results that are easily interpretable. The data set is mined for information that explains the data in new ways and aids further investigation of the data in search for more information. Usually, visualization also takes a large part of the work to fulfill the descriptive goal, to summarize the data in an understandable way.

## 2.4 Data Stream Mining

Data streams are data sources where the contents is not static, data instances arrive incrementally, rapidly, generally unlabeled, and the data streams are potentially infinite in size. These characteristics forces additional consideration when designing the algorithms that are to be used in the domain. The algorithm typically have to manage incremental updates to the model, process instances quickly, and be able to adapt over time as the data in the stream evolves.

The evolving nature of the data streams is commonly depicted as concept drift [32]. It means that the distribution of the data within the data stream is not stationary, it alters as time moves along. These phenomenoms are often divided into four categories, namely sudden, gradual, incremental, and recurring drift. Sudden drifts are when changes happens in a short period of time, can often be seen as the old concept disappears and the new appears instantly. Gradual drift is when a new concept is gradually replacing an older concept, intermittently having data instances from both concepts appearing in the stream. Incremental drift occur when a concept is slowly evolving into a new concept, can almost be seen as the data instances slowly moving in the data space towards its new location. Finally, recurring drifts are when concepts altogether appear and disappear back and forth between eachother.

## 2.5 Outlier Detection

Defining an outlier depend on assumptions regarding the applied detection method and the data structure itself [33]. Although, some definitions are considered general enough to cover most cases. According to Hawkins [34], an outlier is an observation that deviates so much from other observations that it appears it is generated by a different mechanism. Similarly, Barnett et al. [35] defines an outlier as an observation that appears to deviate markedly from other members of the sample it occurs. Generally, an outlier can be seen as an observation that does not adhere to the norm of the data.

In many applications and data sets, there may exist a significant amount of noise that makes it harder to define a sufficient deviation to identify outliers. Normally, it is the significantly deviating observations that are of interest [36]. The observations that are of lesser deviation, the noise, are usually not of interest.

The identification of outliers can be divided into three major approaches, depending on what is known of the data beforehand [36]. If there exist labels for both normality and abnormality, both types of data points can be modeled, and fully supervised learning can be applied. If there are labels for the normal part of the data, the normal behavior is modeled and used to determine if a new observation is normal or not, which is seen as a semi-supervised learning problem. The approach is the same as if there only exists labels for the abnormal part of the data. Finally, if there are no labels, unsupervised learning methods are applied to identify outliers with the assumption that the normal data is distinguishable from the outliers.



---

## Related Work

---

This chapter presents scientific work closely related to the research conducted in this thesis. We begin with works that have focused on electricity consumption and electricity consumption behaviors. We continue with data stream clustering algorithms and conclude the section with clustering of multiple data streams.

### 3.1 Electricity Consumption Behaviors

Determining behavior in regard to electricity consumption has traditionally been approached on a larger scale, primarily for the use of electricity providers and aiding them in forecasting future consumption needs. The approaches can be broadly divided into three areas [37]; 1) Very Short-Term Load Forecasting (VSTLF), where prediction is minutes to hours ahead, 2) Short-Term Load Forecasting, predicting hours to weeks in the future, and 3) Medium and Long Term Load Forecasting where the models predict usage months to years ahead. For our work, we have focused on studies performed in the VSTLF category.

Different types of regression models have been investigated, including LR [38], SVR [39], and neural networks [37], but also approaches such as exponential smoothing [40]. One important finding in the area is that identifying features that captures consumption trends is of great importance [41, 42]. In these studies, most of the data that is used contains electricity consumption from entire nations or at least larger cities. There you have the consumption of the masses and this tends to follow a strict consumption pattern, making it easier to accurately predict. Due to lack of data sources, less focus has been put on individual buildings or households.

With the introduction of smart meters [43], electricity consumption

### 3. RELATED WORK

---

meters that allow remote real time measurements, data can now be collected from office buildings [44], houses [45], and other smaller buildings. Chou and Telaga [44] show that it is possible to detect abnormalities of consumption in real time in an office building, using both clustering analysis and regression. Likewise, Zhang et al. [45] analyze household electricity consumption data to detect abnormal days in a similar fashion. C. Chalmers et al. [46] compare households to each other to identify abnormal consumption patterns. T. Chen et al. [47] create customer profile groups to both determine what type of weekly behavior each user has for each year and uses the groups to detect if a user has changed their consumption behavior compared to last year. They mix regression and clustering for two types of detection. Regression is used to detect changes in terms of how much electricity is consumed and clustering for how the electricity is used, i.e. the shape. Each individual user is profiled into different consumer groups based on their type of consumption. Over time, if a user is profiled into a new group, they determine that the behavior has changed.

## 3.2 Clustering Streaming Data

The limitations present in data stream mining makes the act of data clustering a considerably more difficult problem. Data arrives in such large quantities and rapid pace, making it impossible to keep the data in main memory [3, 4]. A simple approach to cluster streaming data would be to use a conventional algorithm, such as  $k$ -means [48] or DBSCAN [49], and re-cluster the clustering solution after each data object arrives in the stream. This is, however, unfeasible due to time constraints and the resources needed by the algorithms [50, 51].

The first algorithm applicable in this setting was proposed by T. Zhang et al. [52], who provided a two-phase algorithm intended to cluster large databases, named BIRCH. BIRCH divides its work into an online and offline phase, where the online phase summarizes the stream to create cluster feature vectors (CF-vector) using a CF-Tree. Each CF-vector consists of three features; the number of objects  $n$ , linear sum of data values  $LS$ , and sum of the squared data values  $SS$ . All features have the additivity property, meaning they are easy to update both when new data objects are added to the CF-vector and when CF-vectors are merged. When a new data object arrives the CF-Tree is traversed to find the closest CF-vector and then

added to it. All CF-vectors present in the CF-Tree are then clustered using  $k$ -medoids to gain the resulting clustering solution.

The two-phase approach to clustering have since been used extensively(CITE ALL HERE), particularly after the introduction of CluStream [53]. CluStream extended the CF-vectors to let CluStream take recency into account, by incorporating the timestamps of the elements, both the linear and squared sum of timestamp values. The extended CF-vector was named micro-clusters. CluStream used the timestamps to determine if that micro-cluster is longer relevant for the algorithm to keep track of. If the average timestamp become lower than a threshold the micro-cluster is simply removed. Snapshots of the micro-cluster solution was also stored on disk, which allow to see how the stream and its clusters evolve over time. The macro-cluster solution created by clustering the micro-clusters in the offline-phase using a modified variant of  $k$ -means. Other variants of offline clustering have appeared, such as density based DBSCAN ...

Y. Chen et al. [54] followed the same principles as CluStream but used a grid-based approach to the online-phase. Instead of using distance-based metrics and creating micro-clusters, these approaches divide the data space into a cell grid. Data is assigned to the corresponding cells upon arrival and depending on the weight and density of each cell, the cells are deemed differently. These cells can be compared to micro-clusters, but these approaches allow the adjacent cells to be grouped directly to create macro-clusters instead of having an offline component cluster the micro-clusters.

Currently, DBStream [55] is considered to be the best-performing algorithm [56]. DBStream's prominence is based on identifying the shared densities between micro-clusters, meaning the number of points that lie within two micro-clusters. This helps determine if micro-clusters belong to the same macro-cluster. In an empirical review of data stream clustering algorithms, Carnein et al. [56] has shown that DBStream outperforms the other algorithms included in the review in most settings. It is however sensitive to the order of inserted elements and has a larger number of parameters that require tuning.

There has also been proposed methods that do not conform to the two-phase clustering. Lughofe [57] directly create the resulting clustering solution and incrementally updates when new data objects arrive in the

stream. To manage the dynamics of a data stream, specific split and merge criterions are defined. Another approach to clustering data streams is to divide the stream into segments and cluster them individually. Guha et al. [58] proposed the algorithm Stream which clusters each segment using  $k$ -median. The resulting cluster centers from the segment are added into a buckets representing a prototype array. When the array is full,  $k$ -medians is run upon the array to produce medians of medians to represent the stream. Similarly, StreamKM++ and the subsequent [59] samples a subset of the segment and provides a clustering solution based on that subset which is then stored in buffers. When a new segment is clustered, the coresets in the buffer are merged together. Finally, Ailon et al. [60] and Boeva et al. [61] provide algorithms that cluster the next segment by combining the new data segment with the clustering solution from the previous segment.

### 3.3 Clustering Multiple Data Streams

Shifting to multiple data streams, we can approach the problem from a few directions. The first is when the actual streams themselves are to be clustered [62–65]. A typical application is using electricity consumption from households to create consumer groups for the provider to use. Another direction comprises of approaches that cluster the contents of the streams, where the multiple streams contain the same type of data but from multiple sources. Here, streams are either aggregated to a single stream which is clustered [66], or individually micro-clustered and then sent to a centralized location for global clustering [67, 68]. In this thesis, however, we are more interested in the direction of having multiple streams that each represents a part of the system, i.e., each stream gives its own description of the overall system. For instance, see a smart home where we have multiple data streams, such as electricity, water, temperature, and so on, they each describe the what is happening in the home from their point of view.

Using multiple data sources separately to create a global model is common in cluster ensembles, namely feature-distributed clustering [69], and multi-view clustering [70], but is yet to have been widely researched for data streams. We have only found one occurrence of approaching the multi stream problem using a multi-view approach, namely Multi-View Stream (MVStream) [71]. MVStream extends a support vector data descriptor to manage the multi view problem by creating a common kernel space for each

data stream to be mapped to. The identified boundaries are then mapped back to the original kernel space for the stream and the instances are then clustered accordingly.



---

# Methodology

---

This section intends to shed light on the methodology used in this thesis. We start by providing a distinction between science and engineering, and how applied computer science relates to these two. Then, we present the research methodology used in this thesis followed by the used datasets and evaluation. At the end of the section, we provide the validity threats to the results and conclusions that we can draw from this thesis.

## 4.1 The Scientific Method and Engineering Design Process

The distinction between science and engineering is of importance to this thesis as it consists mainly of applied computer science research. Generally, science has throughout the years been seen as identifying general theories that intend to explain the world [72]. Engineering instead relies on using known knowledge and technology to produce solutions tailored to a specific problem, that can be validated to be robust against errors [73]. Applied research tends to lie close to the middle of these terms, and could almost be seen as an intersection of the two.

Applied computer science, which this thesis consists of is usually guided by engineering. The initial process in applied research involves finding a solution to a specific problem. However, rather than providing a solution of known technologies, applied research identifies the knowledge gap to produce the science needed to solve the problem.

## 4.2 Research Methodology

The philosophy of science has been discussed in great detail over many decades and this thesis has no intent of continuing or adding to the discussion. Instead, we only provide a brief introduction to scientific methodology and how we applied it in our research.

New Experimentalism [72], as described by Deborah Mayo, relies on conducting experiments to then derive theory from. In contrast to the more traditional research methodologies, Inductivism and Falsification where general theories and scientific knowledge is derived from observations and facts, the backbone of New Experimentalism is conducting experiments and derive knowledge from their results. The derived theories does, however, require the results to be severely tested, which leads to more experiments which would lead to failed experiments if the theory was false. When errors occur in the experiments, the theories are not discarded as in Falsification, but rather they are embraced to further develop the theories and experiments.

This thesis started from an engineering point of view, intending to identify there was not much research done neither with the data granularity of 1 minute measurements, but especially not on the household level. The knowledge we derived we then used to create the next study, and so on and so forth. This is as close to new experimentalism as you can get.

However, for the final study we deviate from this methodology as we approach it qualitative by surveying the literature. Throughout our experimenting we identified that it was not clear how evaluation was performed in the domain, so we surveyed it to understand the current landscape and draw conclusions based on these.

## 4.3 Data sets

In PAPER I we used a data set containing electricity consumption data gathered at 1 minute intervals. The data was gathered from 20 individual households, during the period 2017-02-27 to 2017-04-01. Out of the 20 households, 3 were discarded for further study due to missing values.

Similarly, in PAPERS II-IV we also used an electricity consumption dataset on an individual household level. It was collected from 2014-01-13 to 2015-04-13 from 9909 households with varying degrees of both quality

and collection length. Out of the 9909 households, the 10 with the largest amount of data were chosen for further study. These 10 households had data from 345 to 353 days of electricity consumption usage.

In PAPER V we used four datasets, namely S1, Covertype, Domestic Electrical Load Metering, Hourly Data (DELMH), and a selfmade synthetic dataset. The S1 dataset [74] is a 2-dimensional synthetic dataset consisting of 5000 data objects distributed in 15 stationary clusters in distinct locations. Covertype [75] is a dataset consisting of 581012 data objects with 54 cartographic features that describe forest images. DELMH is a real world dataset containing hourly electricity consumption data from individual households in South Africa [76]. In total, there are 3341726 data instances collected during the period from 1994 to 2014. Both the quality and collection length for the households varied significantly and the chosen household, which had the largest number of measurements, consisted only of 496 days of electricity consumption.

The selfmade synthetic dataset consisted of spherical clusters and contained both constantly drifting clusters and both appearing and disappearing clusters. It was made using a radial basis function generator (RBFGenerator) based on the implementations available at MOA [77] and scikit-multiflow [78]. Both RBFGenerators mentioned above contained issues, MOA could not export the dataset in certain cases and the scikit-flow lacked functionality, so we created our own with those issues sorted out. The dataset is an evolving data stream of 2-dimensional data with 10000 data instances.

Finally, in PAPER VI we continued using our RBFGenerator to create datasets of 3 streams of 2-dimensional data and 12 streams of 8 dimensional data. Additionally, we used the Almanac of minutely power dataset version 2 (AMPds2) [79]. AMPds2 is a real world dataset of a single household's electricity, water, and gas consumption, and hourly weather data from a nearby weather station. The data was collected in Canada during the period from April 2012 to May 2014.

We also want to provide bring up any ethical aspects of the real world datasets that have been used in these studies. All the electricity consumption datasets are anonymized, the DELMH and AMPds2 datasets are anonymized by their authors, and the datasets used in Papers I-IV we removed all data that was not the actual consumption. However, the data used in Papers I-IV

is not for public domain either, thus, there should be no ethical concerns.

## 4.4 Evaluation Measures

In this thesis, we have explored both regression and clustering analysis to model user behavior. These models require different evaluation measures to assess their quality. In PAPER I, we used Mean Absolute Percentage Error (MAPE) [80] to validate the regression models. MAPE assesses how much relative error there is between the actual value and the predicted value of the regression model. MAPE is defined as

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \times 100, \quad (4.1)$$

where  $A_t$  is the actual value at time  $t$ ,  $F_t$  is the predicted value at time  $t$ , and  $n$  is the number of measurements. MAPE provides a score between 0 and 100, where a lower score is better.

Two main categories of validation measures exist for clustering solutions, internal and external measures. Internal validation measures identify how well the clustering solution is by assessing the internal structure of the solution. External measures, on the other hand, validates the result by measuring against the ground truth [81].

In PAPERS II-VI we applied Silhouette Index (SI) [82] to evaluate the resulting clustering solutions. SI measures compactness of individual clusters and the separation between clusters of any clustering solution  $C = C_1, \dots, C_k$ . The average SI for an entire clustering solution is calculated as

$$\text{SI}(C) = \frac{1}{n} \sum_{i=1}^n (b_i - a_i) / \max\{a_i, b_i\}, \quad (4.2)$$

where  $a_i$  is the average distance from object  $i$  to the other objects in its cluster,  $b_i$  is the minimum average distance from  $i$  to the objects of the other clusters, and  $n$  is the number of objects in the data set. The value is in the range [-1, 1]. Values closer to 1 indicate the solution is clustered well, a value closer to -1 shows that there are misplaced objects, and values close to 0 means there are objects that can be placed in multiple clusters.

Connectivity [83], used in PAPER II-IV, measures how well each object is connected to the other objects within the same cluster. Connectivity is

measured by identifying how many of the nearest neighbors belong to the same cluster and is defined as

$$\text{Co}(\mathbf{C}) = \sum_{i=1}^m \sum_{j=1}^{n_r} \chi_{im_{ij}}, \quad (4.3)$$

where  $m_{ij}$  is the  $j$ th nearest neighbour of object  $i$ , and  $\chi_{im_{ij}}$  is 0 if  $i$  and  $m_{ij}$  belong to the same cluster and  $1/j$  if they are not. Connectivity provides a value between 0 and  $\infty$ , where a low value indicate a good clustering solution.

PAPERS II-IV apply the Average Intra-Cluster Distance (IC-av) [84] which, similarly to SI, provides a measure of how compact the produced clusters are without assuming a spherical shape of the clusters. A Minimum Spanning Tree (MST) is created of the distances between the objects in the data set and the MST edges are used to determine the compactness of the clusters. For a particular clustering solution  $C = C_1, \dots, C_k$ , the *IC-av* is defined as

$$\text{IC-av}(\mathbf{C}) = \sum_{r=1}^k \frac{1}{n} \sum_{i,j \in C_r} d_{ij}^2, \quad (4.4)$$

where  $n$  is the number of objects in cluster  $C_r$  ( $r = 1, 2, \dots, k$ ) and  $d_{ij}$  is maximum edge distance which represents the longest edge in the path joining objects  $i$  and  $j$  in the MST. IC-av produces a value between 0 and  $\infty$ . Low values indicate a good clustering solution.

In PAPER V, we also apply the Jaccard Index (JI) [85] measure to compare the resulting clustering solutions against a ground truth. Given two clustering solutions produced from the same dataset,  $A$  and  $B$ , we define JI between  $A$  and  $B$  as follows:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (4.5)$$

where  $|A \cap B|$  is the number of data points with the same class in the same clusters in  $A$  and  $B$ , and  $|A \cup B|$  is the total number of data points in the same clusters in  $A$  and  $B$ . JI ranges from 0 to 1, where a higher value indicates a higher similarity between the clustering solutions.

The  $F_1$  measure [86], used in PAPERS V-VI, is a commonly used metric to capture the accuracy of a clustering solution. The harmonic mean of the

#### 4. METHODOLOGY

---

precision and recall of the produced clustering solution is calculated and compared against the ground truth. It can be defined as:

$$F_1(C_i) = \frac{2|C_i \cap C_i^d|}{|C_i| + |C_i^d|}, \quad (4.6)$$

where  $C_i$  is the cluster containing the maximum number of objects from  $C_i^d$ .

To evaluate the overall  $F_1$  score for the clustering solution  $C$ , two common approaches are used, micro and macro average. The macro average sees all classes as equal while the micro average corrects the score by each individual class's frequency. The macro  $F_1$  is defined as:

$$F_1(C) = \frac{1}{l} \sum_{i=1}^l F(C_i). \quad (4.7)$$

Finally, in PAPER VI both Adjusted Rand Index (ARI) [87] and Adjusted Mutual Information (AMI) [88] were applied. ARI is the corrected for chance version of the Rand Index and is commonly used to compare against the ground truth clustering by measuring the similarity between the two clustering solutions. Using a contingency table, the overlap between the clustering solutions  $X$  and  $Y$  is summarized by calculating the number of instances  $n_{ij}$  in common between  $X_i$  and  $Y_j$ .

	$Y_0$	$Y_1$	$\dots$	$Y_j$	<i>sums</i>
$X_1$	$n_{11}$	$n_{12}$	$\dots$	$n_{1j}$	$a_1$
$X_2$	$n_{21}$	$n_{22}$	$\dots$	$n_{2j}$	$a_2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$X_i$	$n_{i1}$	$n_{i2}$	$\dots$	$n_{ij}$	$a_i$
<i>sums</i>	$b_1$	$b_2$	$\dots$	$b_j$	

The sums  $b_1, b_2, \dots, b_j$  and  $a_1, a_2, \dots, a_i$  are calculated as the sum of the number of instances in common, for instance  $b_1 = \sum_{k=0}^i n_{k1}$ . The sums  $a_{ij}$  and  $b_{ij}$  are used in conjunction with the number of instances in common  $n_{ij}$  to calculate the ARI as follows:

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}. \quad (4.8)$$

AMI is the corrected for change version of Mutual Information index and is, like ARI, used to compare against ground truth clustering to measure their similarities. AMI can be defined as:

$$\text{AMI} = \frac{MI - E[MI]}{\text{avg}(H(U), H(V)) - E[MI]}, \quad (4.9)$$

where  $MI$  is the mutual information between  $U$  and  $V$ ,  $E[MI]$  is the expected mutual information, and  $H(U)$  and  $H(V)$  is the amount of uncertainty for partition  $U$  and  $V$  respectively.

## 4.5 Validity Threats

In this section we pose the validity threats of the results in this thesis which should be considered, namely *Internal*, *External*, *Construct*, and *Conclusion* validity threats and how minimize their presence.

*Internal validity* concerns the experimental procedures. The internal validity ensures that the observed outcome is due to a causal relationship to the treatment, not an uncontrolling factor [89–91]. Most of the threats to internal validity do not apply to this thesis, due to the nature of the research. However, the threat of selection bias is present which often can be remedied by random sampling [91]. In PAPER I we perform cross-validation to minimize the threat and in PAPER VII it is not a concern as it is a survey of existing literature. Using cross validation is not applicable for PAPERS II-VI due to the nature of the research. We are intending for the modeling of continuously evolving behavior in a data stream and by incorporating cross validation we disrupt the natural development of the stream.

*External validity* refers to the generalization of the experiment results. Providing a positive outcome in an experimental setting does not ensure the same outcome in a different setting [89–91].

The data sets used can threaten the generalizability of the results. Our methods are data-driven and learn about the data. The quality of the data that we have used, both the synthetic and real-world data, can impact the generalizability of the evaluated and proposed approaches. Additionally, the limited number of datasets might not be enough to warrant for generalizability to all real world application. In total we have analyzed 29 households, where 27 resides in Sweden, 1 in South Africa, and 1 in Canada. There

#### 4. METHODOLOGY

---

might be large discrepancies in the consumption patterns between countries which could impact the generalizability of the results.

*Construct validity* indicates how the results of a study generalizes to the concepts and theories behind it. The construct validity threats come from inadequate definitions and measurement of variables. [89–91].

Implementing the experiments and identifying the appropriate parameters provide possible threats to the correctness and reliability. In PAPER I we perform parameter tuning on several regression models by evaluating the resulting models from those parameters empirically using the mean average percentage error (MAPE). Likewise, in PAPERS II-VI we apply the partitioning algorithm  $k$ -medoids which requires the parameter  $k$  to be defined beforehand. For the labeled datasets we initialized the algorithm with the ground truth, but we empirically investigated values of  $k$  to identify the appropriate value on the unlabeled datasets. We applied CVIs that are commonly used in the domain to evaluate the produced models from varying the  $k$  parameter, see Section 4.4 for the full list of CVIs.

*Conclusion validity*, also referred to as statistical conclusion validity, concerns when the experimenter draws inaccurate inferences from the data and experiments [89–91]. In PAPERS I-VI we investigate algorithms' performance of the modeling of unlabeled datasets and the intention is to provide insights about the data and its behavior.. This raises some threats to the conclusion validity. There is no ground truth to evaluate against, which limits the evaluation we can perform. However, we do standardized measures to evaluate the performance, as seen in Section 4.4. Furthermore, the analysis we perform have gone through peer-review in the publication of the studies which minimizes the threat of inaccuracies in the analysis and conclusions.

## Proposed Data Mining Algorithms and Results

In this chapter we present the results that intend to answer the posed research questions. The first sections correspond to each research question, where main findings and a discussion is located within the individual sections. In the final section we discuss the present validity threats for the thesis.

### 5.1 Modeling User/System Behavior

We approached the first research question regarding the modeling of user/system behavior of single data streams using two methods. In PAPER I, we investigated the use of regression models to model single data streams, using the produced model to predict the future data. The investigation evaluated several aspects of building the regression model, including, data granularity, feature selection, and ability to detect abnormal behaviors. In PAPER II, we instead investigated the use of clustering analysis to model single data streams. Below we detail each method and show their individual contributions. At the end of the section we provide the conclusions drawn according to the first research question.

**Regression** In PAPER I, we analyze electricity consumption data from individual households and investigate the possibility of modeling the household behavior. Additionally, the dataset is also investigated to identify suitable data granularity and features.

In the data analysis we identify the effect of the dataset's granularity. The electricity consumption data was collected with a 1 minute interval, meaning we have 60 data points for each hour of data. With this granularity it was clear that the consumption behaviors were hard to visually identify.

## 5. PROPOSED DATA MINING ALGORITHMS AND RESULTS

---

The data was therefore aggregated into 10 minute, 30 minute, and 60 minute intervals as well. The consumption patterns are much more visually present in these datasets, hinting towards the granularity to be an important aspect of building the regression models.

To identify suitable features for the regression model we calculate the auto-correlation of each stream, for all the granularity datasets. Similarly to the visual inspection of the data streams, the auto-correlation were consistently higher scoring in the data sets with higher granularity. The auto-correlation of the streams show presence of both daily and weekly consumption behaviors in the data. Feature sets that represent daily and weekly behaviors are created based on these results and evaluated against each other in terms of prediction accuracy. Interestingly, the feature sets containing fewer features that focused on the daily behaviors, i.e., consumption from just before the predicted value, produced better prediction models.

**Clustering** In PAPER II the research question is approached by performing cluster analysis on the same dataset to model the user/system behavior. The  $k$ -medoids algorithm is applied to determine a set of user/system behaviors present in the data. The centroids of the resulting clustering solution are then used as exemplars of the user/system behaviors.

Two dissimilarity measures, ED and DTW, are evaluated to identify their differences in the resulting clusters. ED's resulting clusters consists of more coherent clusters, compared to the clusters generated using DTW. The content of each major ED cluster align with the chosen signature and all consumption peaks occur at similar points in time. DTW, on the other hand, creates fewer clusters, and the content of each cluster have less coherence. The difference of the clusters between ED and DTW is to be expected, as DTW warps the data to try and align them as best as possible. However, the amount of warping allowed is an important aspect and a stricter amount is necessary in the continuing studies.

The distribution of weekdays between clusters is also a difference between the two dissimilarity measures. ED creates a clear distinction between working days and holidays, where three clusters consists majorly of saturdays and sundays. On the contrary, DTW creates fairly monotonic clusters that have an even distribution of the weekdays in each cluster.

In this section, we have shown two directions of how modeling of user/system behavior of streaming data can be performed, using regression and clustering.

Specifically, in PAPER I we showed that household electricity consumption data is possible to model using regression models. Data granularity and features to include was also shown to be an important aspect of building the model.

In PAPER II, we have shown clustering is a viable option to model the user/system behavior of a household's electricity consumption. Using ED as dissimilarity measure seems to be better suited for the task, with both a clearer distinction between cluster and distribution of weekdays compared to DTW.

## 5.2 Modeling Single Evolving Data Streams

We approach the second research question by analyzing the clustering results from two segments of evolving data streams PAPER III. The paper addresses the problem by mining the stream for behavior changes, by clustering the stream in segments and comparing the segments' clustering solutions. Based on these results, we design a sequential clustering algorithm that continuously clusters an evolving data stream in PAPER IV and finalized in PAPER V.

**Identifying changes** In PAPER III we divide a data stream into two segments, the first representing the historical data and the other the newly arrived segment. The clustering solution gathered from the historical data segment is used to seed the clustering of the new data segment, by initializing the new clustering with centroids from the historical clustering solution.

This methodology builds upon the assumption that the same, or similar, behavior is present continuously in the evolving data stream. If the same clusters remain in the new segment there has been no changes. On the other hand, if there clusters have changed, disappeared, or emerged in comparison to the previous segment, we identify changes in the behavior of the stream.

The results from the study show potential for an algorithm that repeatedly performs the seeding procedure.

**Sequential Clustering** Building upon the results from the previous study, PAPERS IV and V define a sequential clustering algorithm, named EvolveCluster, that continuously clusters segments of a data stream using centroids from the previous segment's clustering solution as initialization seeds.

EvolveCluster divides a stream into segments where each segment is clustered individually. By doing so, we can analyze and monitor how the clustering solutions alter over time, showing how the behavior in the stream is evolving. The disappearing, merging, or creation of new clusters in each segment are inherently handled by the clustering itself, though splitting of the clusters are handled by a separate splitting mechanism that applies a trial-and-error approach to identify if any clusters are suitable for splitting. An illustration of the general procedure for EvolveCluster is found in Figure 5.1.

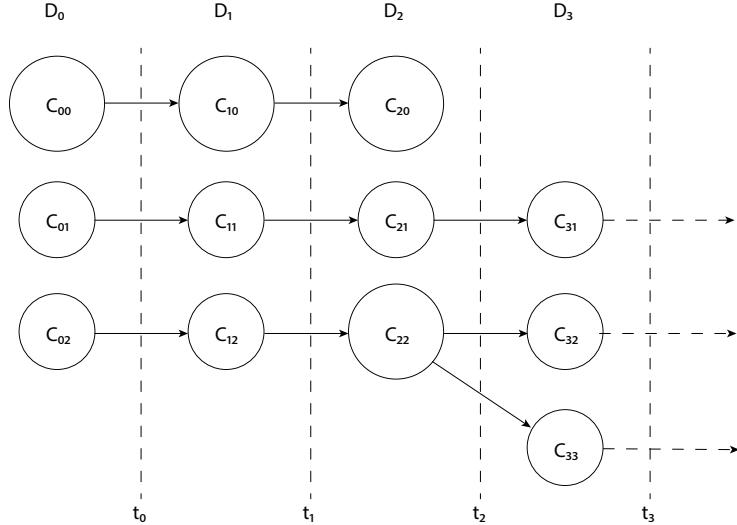


Figure 5.1: A schematic illustration of the proposed EvolveCluster.  $D_i$  represents data segments at time  $t_i$ ,  $C_{ij}$  represents clusters in clustering solutions  $C_i$  of data segments  $D_i$ , and  $t$  represents individual time segments.

EvolveCluster was evaluated against two algorithms, PivotBiCluster and Evolutionary Split-Merge Clustering, that similarly divides a stream into segments and uses prior knowledge from the previous segment to cluster the newly arrived segment. Experiments were designed to showcase the capabilities of all algorithms in clustering both highly dynamic and less

dynamic streams. PAPER V shows that EvolveCluster manages both these types of streams, while performing better in the less dynamic streams. This is to no surprise, as EvolveCluster assumes that no major change has happened since the last segment with its injection of centroids from the previous data segment. If there is no evolving behavior present, it is likely that EvolveCluster will place the data points correctly in the initial partitioning and no refinement is required.

EvolveCluster relies heavily on the splitting mechanism when higher dynamics are present. After the initial partitioning of a segment, in a high dynamic stream, there will be misplaced data points and too much separation within the clusters. The trial-and-error splitting approach proposed in the algorithm can be computationally exhaustive, as all clusters are evaluated for a possible split. If a split occurs, then the splitting procedure starts over and re-evaluates all clusters for splits again. However, this is part of tuning the parameter  $\tau$  which specifies the information gain needed by a split for it to be included in the clustering solution. It is important to remember that the intention for EvolveCluster is to operate on less dynamic streams, where fewer splits and merges occur.

One of the benefits of EvolveClusters design is the possibilities to trace how the behavior changes over time. As we use previous centroids as initialization for the current segment, we can simply compare the centroids over time and see how the stream and its clusters evolve.

## 5.3 Modeling Multiple Evolving Data Streams

In PAPER VI we approach this research question by designing a multi stream version of EvolveCluster. The new algorithm, MultiStream EvolveCluster (MS-EC), contains a EvolveCluster module per data stream in the system. After each segment has been clustered by the EvolveCluster modules the labels from the clustering solution is sent to a centralized location where a consensus clustering is performed. This approach provides both individual clusterings of every single stream and the consensus clustering offers possibilities for different combinations of the streams. A general illustration is present in Figure 5.2 below.

Approaching the problem of multi stream clustering by viewing each stream as its individual view, clustering them separately, and then using their

## 5. PROPOSED DATA MINING ALGORITHMS AND RESULTS

---

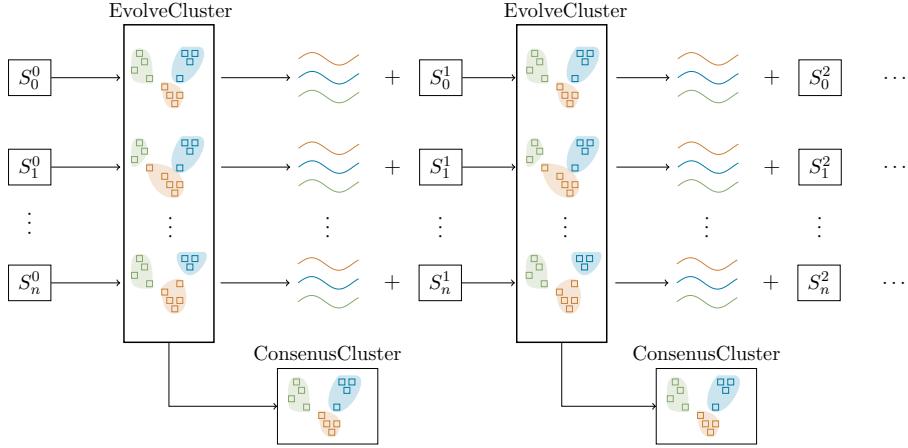


Figure 5.2: *MultiStreamEvolveClusters* general procedure while clustering a set of data streams.  $S$  is a set of  $n$  streams, i.e.  $S = \{S_0, \dots, S_n\}$ . Each stream is divided into equal-sized segments, so  $S_i^j$  is the  $j$ th segment of stream  $S_i$ . Each stream is clustered separately, and after a segment is clustered, the clusterings are combined to determine a consensus.

clustering solutions to create a global clustering solution provides certain benefits. First, the privacy of the data present in the streams stays intact. The data is never sent to the centralized server, only the cluster labels themselves. Secondly, the computational efforts can be distributed onto computers in the sensors vicinity, or even the sensors themselves. Third, the resulting algorithm becomes data agnostic. Each stream's clustering module can be designed independently from each other which allow for optimization of both underlying clustering algorithm, dissimilarity measure, and data granularity individually. The only requirement is that each data object represents the same amount of time in total. For instance, in the experiments on the AMPds2 dataset, we create daily profiles of 4 hour measurements for the electricity consumption while we have 6 hour measurements for the water consumption. The profile for the electricity contains 6 data points in the profile while the water profile contains 4.

Three methods were evaluated to perform the consensus clustering, namely *Majority Vote*, *Hypergraph Partitioning*, and *Markov clustering*. Additionally, these three methods were compared against a baseline, a single EvolveCluster module was applied to a merged data stream consisting of the multiple streams.

We show that MS-EC produce better quality clustering solutions on synthetic datasets compared to the baseline. Two synthetic and one real world dataset was used for the evaluation. The synthetic datasets consisted of 3 streams of 2-dimensional data and 12 streams of 8-dimensional data. Already in the 2-dimensional dataset all MS-EC variants provide better quality clustering solutions compared to the baseline, where Majority Vote produces perfect scores of  $F_1$ , ARI, AMI and a 0.96 SI score. The baseline instead produces negative SI scores and  $F_1$ , AMI, and ARI scores close to 0. Similarly in the 8-dimensional dataset, MS-EC manages to produce higher quality clustering solutions when using both Majority Vote and Hypergraph Partitioning compared to the baseline and Markov clustering variant of MS-EC. It is clear that MS-ECs division of work is successful compared to merging multiple streams into one and clustering that.

The real world dataset, consisting of data streams containing electricity consumption, water consumption, gas consumption, and outdoor temperature, provided a tougher challenge for MS-EC, even losing to the baseline algorithm. This was due to the difficulties in clustering the individual streams with the EvolveCluster modules, the electricity and water stream was shown to be easier to cluster while the gas and, especially, the weather streams were harder. Therefore, we investigated multiple combinations of the individual streams and removing the weather stream from the consensus clustering provided a better consensus clustering for all three versions of MS-EC, albeit still lesser scoring than the baseline algorithm for all four streams.

## 5.4 Evaluating Data Stream Clustering Algorithms

PAPER VII During our investigation of the previous research questions, it became clear that the evaluation of clustering algorithms for streaming data included some varying techniques. No real consensus was present in the literature, which we thought to investigate further. Therefore, in PAPER VII we review the literature containing data stream clustering algorithms, including previous surveys of the domain, to map out which and how cluster evaluation techniques are used.

Our review identifies only one CVI specifically designed for evaluation of data stream clustering algorithms. Namely, the external validation measure

## 5. PROPOSED DATA MINING ALGORITHMS AND RESULTS

---

Cluster Mapping Measure (CMM). CMM quantifies the aspects of importance for data stream clustering algorithms, such as recentness... No other evaluation measures have been specifically designed for these algorithms and the evaluation measures available from traditional clustering are prevalent in this domain.

The most common CVIs in the literature is purity for labeled datasets and SSQ for non-labeled datasets, . However, in data stream clustering we are often interested in how the *current* behavior is modeled by the algorithm. Data that was present a long time ago has less of an impact than the current model. The common approach to filter out the historical data is to use CVIs within a sliding or landmark window. Thus, only the  $n$  most recent data points are included when calculating CVIs while the algorithm still has the ability to model all the points seen so far.

We identified that the online component of the algorithms has been the center of attention when evaluating. Less focus is put on the offline component, which is the resulting clustering solution for the practitioner to use. Comparing algorithms becomes an issue when only one part of the algorithm is evaluated. Indeed, if two algorithms that both employ the two-phase clustering scheme then the micro-cluster solution with a greater accuracy should be beneficial for the offline component. However, there might be edge cases where this is not true and could influence the offline component to create more accurate clustering solutions for the lesser performing online component.

We also identified hardships of used data sets, especially in comparing results between algorithms. Many algorithms are designed to handle a specific case and then some general datasets are used, with the likes of KDD-CUP'99 (network intrusion dataset), where some has issues. For instance, the KDD-CUP'99 dataset is commonly used and contains a long period of time only a single cluster exists. This causes many, if not all, algorithms to produce high scoring results, especially so when measures such as purity are used.

### 5.5 Summary

With the assistance of the results presented in this section, we answer the posed research questions below.

**RQ 1:** *How can streaming data be modeled for profiling user/system behavior patterns?*

We have shown that mining for patterns in streaming data can be performed in a high granularity data stream. Using both regression in PAPER I, and clustering analysis in PAPER II, we were able to model single household energy consumption. By doing so, we also identified aspects of the data that is required to take into account. First, the granularity of the data greatly affects both the performance of the algorithms but also the visualization for the practitioner. Secondly, the behaviors that provide the most information to the modelling are the daily behaviors, which our feature-sets created in PAPER I shows. Finally, when the time of behaviors are important ED, or at least a restricted version of DTW, seems more capable to distinguish between the different behaviors in the data.

**RQ 2:** *How can we model and monitor user/system behavior in singular evolving data streams?*

The modeling of evolving data streams we have shown to be tackled by the proposal of EvolveCluster. In PAPER III we produced the groundwork of EvolveCluster by performing change detection of a data stream, by introducing the use of historical data (i.e. centroids) to influence the clustering of the new data. This approach was formalized in PAPER IV, where a continuous repetition of the seeding was performed for an endless stream. With the addition of a specific split mechanism we proposed EvolveCluster in PAPER V. Using the previous centroids, EvolveCluster assumes and capitalizes on the stream staying moderately stable over time. When changes occur, they are easy to identify by comparing the centroids from the segments of interest. Furthermore, the use of prior knowledge also decreases the time for the underlying clustering algorithm to converge.

**RQ 3:** *How can we take advantage of multiple data streams to model the behavior of a user/system?*

We addressed this research question in PAPER VI by proposing a clustering algorithm using a distributed computing mindset. By clustering this with a federated learning approach, we clustered each stream individually and then merge the results together to keep privacy intact in addition to lowering the computational costs and sent traffic. We show that this research problem

## 5. PROPOSED DATA MINING ALGORITHMS AND RESULTS

---

is of interest and a trickier problem than first meets the eye. Often data streams are joined together to have a centralized computational server that merges the data together. Instead, by distributing the computation and only using labels as the input to the centralized location we let each stream live its own life which allow us to both monitor the streams separately or combinations thereof.

**RQ 4:** *Which cluster evaluation techniques are suitable to be used to evaluate data stream clustering algorithms?*

In this thesis, we have proposed and evaluated several clustering algorithms for use in clustering data streams which led us to identify a need for mapping current evaluation techniques used in the domain. In PAPER VII, we performed a survey of the current landscape of evaluation for data stream clustering algorithms. We identified that performance, in terms of accuracy and run time, is the focal point of the domain. Sliding and landmark windows are applied to the evaluation to make CVIs such as purity and SSQ applicable. The offline component, producing the resulting macro-clusters, receives less attention. Thus, aspects such as cluster compactness, separation, and connectedness are not assessed on the macro-clustering solutions.

- A concluding paragraph about the bigger picture of the results is to be filled in.

---

## Conclusions and Future Work

---

In this thesis, we have explored the modeling of user/system behaviors present in evolving data streams. Specifically, we have designed clustering algorithms capable of modeling and mining the evolving behavior in both single and multiple data streams. First, we identify how data streams could be mined to create profiles of the user/system behavior. We use the gained knowledge to develop a clustering algorithm capable of modeling evolving data streams, namely EvolveCluster. A common thought to improve models is to increase the amount of data, both in terms of quantity and adding sources. We instead aimed for creating multiple models, one for each data stream, and designed MultiStream EvolveCluster (MS-EC). MS-EC distributes the clustering to each individual stream to let each stream run its course and only submit the clustering solutions cluster labels when creating the global clustering solution. Finally, through our research we identified a lack of methodology for evaluating the proficiency of data stream clustering algorithms. To fully grasp the extent of the evaluation methodologies in the domain, we performed a survey to map how the evaluation, both in terms of methodology and the evaluation measures used.

For our future work, we see many directions to venture further to. First, the creation of MS-EC provided an interesting problem statement where multiple streams of a user/system, each containing unique information, are present. With the restriction of keeping the privacy of the data intact and minimizing communication, we intend to further research the problem and investigate if pure data stream clustering algorithms could be useful for the individual streams. Moreover, the modularity of MS-EC gives possibilities of many combinations of underlying clustering algorithms and application domains. One of particular interest is to investigate the possibilities to use non-spherical clustering algorithms to possibly improve the capabilities of MS-EC.

## 6. CONCLUSIONS AND FUTURE WORK

---

Furthermore, the use of  $k$ -medoids in our algorithms limits our resulting clustering solution in the sense that only spherical clusters are produced. An interesting way forward, for both EvolveCluster and subsequently MS-EC, would be to investigate the possibilities to create non-spherical clusters, with an underlying clustering algorithm such as DBSCAN [49] or the like.

---

## Bibliography

---

- [1] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu. “Data stream analysis: Foundations, major tasks and tools”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.3 (2021), e1405.
- [2] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, et al. “Open challenges for data stream mining research”. In: *ACM SIGKDD explorations newsletter* 16.1 (2014), pp. 1–10.
- [3] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
- [4] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. “Moa: Massive online analysis, a framework for stream classification and clustering”. In: *Proceedings of the First Workshop on Applications of Pattern Analysis*. PMLR. 2010, pp. 44–50.
- [5] S. Bickel and T. Scheffer. “Multi-view clustering.” In: *IEEE Int. Conf. Data Min. 2004*. Vol. 4. 2004, pp. 19–26.
- [6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [7] A. L. Samuel. “Some Studies in machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* (1959), pp. 210–229.
- [8] M. Campbell, A. J. Hoane Jr, and F. Hsu. “Deep blue”. In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83.
- [9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), p. 484.

- [10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. “Mastering the game of go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.
- [11] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [12] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [13] P. Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.
- [14] A. C. Rencher and G. B. Schaalje. *Linear models in statistics*. John Wiley & Sons, 2008.
- [15] C. Cortes and V. Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.
- [16] S. Avidan. “Support vector tracking”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.8 (2004), pp. 1064–1072.
- [17] O. Chapelle, P. Haffner, and V. N. Vapnik. “Support vector machines for histogram-based image classification”. In: *IEEE transactions on Neural Networks* 10.5 (1999), pp. 1055–1064.
- [18] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. “Support vector regression machines”. In: *Advances in neural information processing systems*. 1997, pp. 155–161.
- [19] J. C. Loehlin. *Latent variable models: An introduction to factor, path, and structural analysis*. Lawrence Erlbaum Associates, Inc, 1987.
- [20] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.
- [21] P. Berkhin. “A survey of clustering data mining techniques”. In: *Grouping multidimensional data*. Springer, 2006, pp. 25–71.
- [22] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silberman, and A. Y. Wu. “A local search approximation algorithm for k-means clustering”. In: *Computational Geometry* 28.2-3 (2004), pp. 89–112.
- [23] V. I. Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

- [24] R. W. Hamming. “Error detecting and error correcting codes”. In: *The Bell system technical journal* 29.2 (1950), pp. 147–160.
- [25] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. “Experimental comparison of representation methods and distance measures for time series data”. In: *Data Mining and Knowledge Discovery* 26.2 (2013), pp. 275–309.
- [26] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. “Querying and mining of time series data: experimental comparison of representations and distance measures”. In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552.
- [27] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.
- [28] M. Vlachos, G. Kollios, and D. Gunopoulos. “Discovering similar multidimensional trajectories”. In: *Proceedings 18th international conference on data engineering*. IEEE. 2002, pp. 673–684.
- [29] G. N. Lance and W. T. Williams. “Mixed-Data Classificatory Programs I - Agglomerative Systems”. In: *Australian Computer Journal* 1.1 (1967), pp. 15–20.
- [30] D. J. Hand. “Data Mining”. In: *Encyclopedia of Environmetrics* 2 (2006).
- [31] M. Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.
- [32] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. “Characterizing concept drift”. In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 964–994.
- [33] I. Ben-Gal. “Outlier detection”. In: *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 131–146.
- [34] D. M. Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.
- [35] V. Barnett and T. Lewis. *Outliers in statistical data*. Wiley, 1974.
- [36] C. C. Aggarwal. “Outlier analysis”. In: *Data mining*. Springer. 2015, pp. 237–263.

- [37] H. S. Hippert, C. E. Pedreira, and R. C. Souza. “Neural networks for short-term load forecasting: A review and evaluation”. In: *IEEE Transactions on power systems* 16.1 (2001), pp. 44–55.
- [38] T. Hong, M. Gui, M. E. Baran, and H. L. Willis. “Modeling and forecasting hourly electric load by multiple linear regression with interactions”. In: *Power and Energy Society General Meeting, 2010 IEEE*. IEEE. 2010, pp. 1–8.
- [39] A. Setiawan, I. Koprinska, and V. G. Agelidis. “Very short-term electricity load demand forecasting using support vector regression”. In: *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE. 2009, pp. 2888–2894.
- [40] J. W. Taylor. “An evaluation of methods for very short-term load forecasting using minute-by-minute British data”. In: *International Journal of Forecasting* 24.4 (2008), pp. 645–658.
- [41] I. Koprinska, M. Rana, and V. G. Agelidis. “Correlation and instance based feature selection for electricity load forecasting”. In: *Knowledge-Based Systems* 82 (2015), pp. 29–40.
- [42] I. Drezga and S. Rahman. “Input variable selection for ANN-based short-term load forecasting”. In: *IEEE Transactions on Power Systems* 13.4 (1998), pp. 1238–1244.
- [43] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana. “A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings”. In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), pp. 1460–1495.
- [44] J.-S. Chou and A. S. Telaga. “Real-time detection of anomalous power consumption”. In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 400–411.
- [45] Y. Zhang, W. Chen, and J. Black. “Anomaly detection in premise energy consumption data”. In: *Power and energy society general meeting, 2011 ieee*. IEEE. 2011, pp. 1–8.
- [46] C. Chalmers, W. Hurst, M. Mackay, and P. Fergus. “Profiling Users in the Smart Grid”. In: *The Seventh International Conference on Emerging Networks and Systems Intelligence*. 2015.

- 
- [47] T. Chen, A. Mutanen, P. Järventausta, and H. Koivisto. “Change detection of electric customer behavior based on AMR measurements”. In: *PowerTech, 2015 IEEE Eindhoven*. IEEE. 2015, pp. 1–6.
  - [48] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.
  - [49] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.
  - [50] M. Mousavi, A. A. Bakar, and M. Vakilian. “Data stream clustering algorithms: A review”. In: *Int J Adv Soft Comput Appl* 7.3 (2015), p. 13.
  - [51] A. Zubayoglu and V. Atalay. “Data stream clustering: a review”. In: *Artif Intell Rev* 54 (2021), pp. 1201–1236.
  - [52] T. Zhang, R. Ramakrishnan, and M. Livny. “BIRCH: A new data clustering algorithm and its applications”. In: *Data Mining and Knowledge Discovery* 1.2 (1997), pp. 141–182.
  - [53] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang. “A framework for clustering evolving data streams”. In: *Proceedings 2003 VLDB conference*. Elsevier. 2003, pp. 81–92.
  - [54] Y. Chen and L. Tu. “Density-based clustering for real-time stream data”. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 133–142.
  - [55] M. Hahsler and M. Bolaños. “Clustering data streams based on shared density between micro-clusters”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.6 (2016), pp. 1449–1461.
  - [56] M. Carnein, D. Assenmacher, and H. Trautmann. “An empirical comparison of stream clustering algorithms”. In: *Proceedings of the computing frontiers conference*. 2017, pp. 361–366.
  - [57] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving systems* 3.3 (2012), pp. 135–151.
  - [58] S. Guha and N. Mishra. “Clustering data streams”. In: *Data stream management*. Springer, 2016, pp. 169–187.

- [59] R. Anderson and Y. S. Koh. “StreamXM: An adaptive partitional clustering solution for evolving data streams”. In: *International Conference on Big Data Analytics and Knowledge Discovery*. Springer. 2015, pp. 270–282.
- [60] N. Ailon, N. Avigdor-Elgrabli, E. Liberty, and A. Van Zuylen. “Improved approximation algorithms for bipartite correlation clustering”. In: *SIAM J Comput* 41.5 (2012), pp. 1110–1121.
- [61] V. Boeva, M. Angelova, V. M. Devagiri, and E. Tsiropkova. “Bipartite split-merge evolutionary clustering”. In: *International conference on agents and artificial intelligence*. Springer. 2019, pp. 204–223.
- [62] P. Laurinec and M. Lucká. “Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting”. In: *Data Min. Knowl. Discovery* 33.2 (2019), pp. 413–445.
- [63] K. Gajowniczek, M. Bator, and T. Ząbkowski. “Whole time series data streams clustering: dynamic profiling of the electricity consumption”. In: *Entropy* 22.12 (2020), p. 1414.
- [64] K. Gajowniczek, M. Bator, T. Ząbkowski, A. Orłowski, and C. K. Loo. “Simulation Study on the Electricity Data Streams Time Series Clustering”. In: *Energies* 13.4 (2020), p. 924.
- [65] A. Balzanella and R. Verde. “Histogram-based clustering of multiple data streams”. In: *Knowl. Inf. Syst.* 62.1 (2020), pp. 203–238.
- [66] M.-H. Le-Nguyen et al. “Continuous Health Monitoring of Machinery using Online Clustering on Unlabeled Data Streams”. In: *2022 IEEE Int. Conf. on Big Data*. IEEE. 2022, pp. 1866–1873.
- [67] Z. Razavi Hesabi, T. Sellis, and K. Liao. “DistClusTree: A Framework for Distributed Stream Clustering”. In: *Databases Theory Appl.* Cham: Springer, 2018, pp. 288–299.
- [68] J. S. Challa et al. “Anytime clustering of data streams while handling noise and concept drift”. In: *J. Exp. Theor. Artif. Intell.* 34.3 (2022), pp. 399–429.
- [69] A. Strehl and J. Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. In: *J. Mach. Learn. Res.* 3.Dec (2002), pp. 583–617.

- [70] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Min. Anal.* 1.2 (2018), pp. 83–107.
- [71] L. Huang, C.-D. Wang, H.-Y. Chao, and S. Y. Philip. “MVStream: Multiview data stream clustering”. In: *IEEE Trans. Neural. Netw. Learn. Syst.* 31.9 (2019), pp. 3482–3496.
- [72] A. F. Chalmers. *What is this thing called science?* Hackett Publishing, 2013.
- [73] S. Sheppard, A. Colby, K. Macatangay, and W. Sullivan. “What is engineering practice?” In: *International Journal of Engineering Education* 22.3 (2007), p. 429.
- [74] P. Fränti and O. Virmajoki. “Iterative shrinking method for clustering problems”. In: *Pattern Recognition* 39.5 (2006), pp. 761–765. DOI: [10.1016/j.patcog.2005.09.012](https://doi.org/10.1016/j.patcog.2005.09.012).
- [75] S. Hettich and S. Bay. *The UCI KDD Archive. University of California, Department of Information and Computer Science, Irvine, CA.* 1999. URL: <http://kdd.ics.uci.edu>.
- [76] W. Toussaint. “Domestic Electrical Load Metering, Hourly Data 1994–2014. version 1.” In: <https://www.datafirst.uct.ac.za/dataportal/index.php/catalog/759> (2019).
- [77] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. “MOA: Massive Online Analysis”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 1601–1604. URL: <http://portal.acm.org/citation.cfm?id=1859903>.
- [78] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. “Scikit-multiflow: A multi-output streaming framework”. In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 2915–2914.
- [79] S. Makonin et al. “Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014”. In: *Sci. Data* 3.1 (2016), pp. 1–12.
- [80] P. N. Dawadi, D. J. Cook, and M. Schmitter-Edgecombe. “Modeling patterns of activities using activity curves”. In: *Pervasive and mobile computing* 28 (2016), pp. 51–68.
- [81] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. “On clustering validation techniques”. In: *Journal of intelligent information systems* 17.2-3 (2001), pp. 107–145.

- [82] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
- [83] J. Handl, J. Knowles, and D. B. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.
- [84] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.
- [85] P. Jaccard. “The distribution of the flora in the alpine zone. 1”. In: *New Phytol* 11.2 (1912), pp. 37–50.
- [86] C. J. Van Rijsbergen. *Information Retrieval*. 2nd. Newton, MA. 1979.
- [87] L. Hubert and P. Arabie. “Comparing partitions”. In: *J. Classif.* 2 (1985), pp. 193–218.
- [88] N. X. Vinh, J. Epps, and J. Bailey. “Information theoretic measures for clusterings comparison: is a correction for chance necessary?” In: *Proc. 26th Annu. Int. Conf. Mach. Learn.* 2009, pp. 1073–1080.
- [89] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [90] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Boston: Houghton Mifflin, 2002.
- [91] J. W. Creswell and J. D. Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

# Detection of Residents' Abnormal Behaviour by Analysing Energy Consumption of Individual Households

*Christian Nordahl, Marie Persson, and Håkan Grahn*

*In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 1st International Workshop on AI for Aging, Rehabilitation and Independent Assisted Living (ARIAL), November, 2017, New Orleans, Florida, USA, pp. 729-738.*

## Abstract

As average life expectancy continuously rises, assisting the elderly population with living independently is of great importance. Detecting abnormal behaviour of the elderly living at home is one way to assist the eldercare systems with the increase of the elderly population. In this study, we perform an initial investigation to identify abnormal behaviour of household residents using energy consumption data. We conduct an experiment in two parts, the first to identify a suitable prediction algorithm to model energy consumption behaviour, and the second to detect abnormal behaviour. This approach allows for an initial step for the elderly care that has a low cost, is easily deployable, and is non-intrusive.

## 7.1 Introduction

With an ever improving healthcare system, average life expectancy is continuously rising. By year 2020, projections state that approximately 20% of the world population will be age of 60 or older [1]. A higher life expectancy provides challenges to ageing adults, e.g. limitations in vision, physical activities, and cognitive decline. These limitations also become challenges for the healthcare and eldercare systems, both in terms of scale and economy.

Assisting the elderly population with living independently at home is one way to approach these concerns.

As smart meters are being deployed world wide by electrical providers, there is an opportunity to provide a low cost approach to remote monitoring of residents that is non-intrusive. Smart meters are energy consumption meters that allows for high resolution energy consumption measurements, and the data can be gathered remotely. It has been shown that the use of electrical appliances directly relate to how the residents perform their activities of daily living (ADLs) [2]. We believe that this can also be shown in how the residents adhere to their normal behaviour of energy consumption.

We have collected energy consumption data from 17 different households in Sweden, provided by a Swedish electricity provider. The collected data is measured with an interval of 1 minute. We will investigate the possibility to detect abnormal behaviour of the resident by analysing the household energy consumption.

In this paper we make the following contributions:

- We model the normal energy consumption behaviour of 17 households by applying the principles of Very Short Term Load Forecasting (VSTLF).
- We investigate the possibility of detecting abnormal behaviour utilising the prediction models modelling the normal energy consumption behaviour.

To summarize, we investigate the possibility to use household energy consumption to detect abnormal behaviour of the residents. We believe this could be a possible first step for eldercare systems to monitor individuals remotely with low cost and no intrusion. As the next step following this study, we intend to collect data from a representative test group in collaboration with our local municipality and eldercare.

## 7.2 Background

In this section, we first define two scenarios of abnormal behaviour that are of interest. We then describe the basics of regression based anomaly

detection, followed up by energy consumption forecasting. We conclude this section with a review of previous work related to ours.

### 7.2.1 Abnormal Behaviour

We have identified two scenarios that are of significance to detect in this type of context; S1: increase of energy consumption during late evenings and nights, and S2: lack of energy consumption during mornings.

Scenario S1 is an indication of a phenomenon called *sundowning*, which often is associated with Alzheimer's disease and in other forms of dementia [3]. When the sun is settling down, the subjects affected with sundowning gets more confused, restless, and agitated etc.

Scenario S2 represents the resident staying in bed longer than usual. This could be an indication of the resident being sick and thereby staying longer in bed.

### 7.2.2 Regression Based Anomaly Detection

The first step of applying regression to detect anomalies in time series data is to fit a model to the data. The model, modelling the normal behaviour, is then used to predict values. Predicted values are compared against the observed by calculating the residual. The residual is then used either directly by its magnitude, or with statistical tests to determine if it is an anomaly or not [4]

The anomalies found using the method above are point anomalies, i.e. single data points identified as abnormal. In our future studies we will investigate the use of contextual anomalies.

### 7.2.3 Forecasting Energy Consumption

Forecasting energy consumption has been approached in many different ways, including Support Vector Regression (SVR) [5], Linear Regression (LR) [6], and exponential smoothing [7]. The most popular being Neural Networks (NN), which Hippert et al. [8] have conducted a thorough survey on. In this context, when identifying abnormal behaviour of elderly people, we generally want to identify if something abnormal has happened as soon as possible. Therefore, we focus on approaches from Very Short Term Load Forecasting

(VSTLF), a research area aimed at forecasting energy consumption minutes to hours ahead.

In VSTLF, the main type of features are previous energy consumptions during the same time on earlier days. These are chosen to capture the daily and weekly trends. It has been shown that fully capturing these trends is important for accurate prediction models [9, 10].

The average use of VSTLF is for electricity providers to estimate how much energy is needed in an area. Studies have mainly been focusing on energy consumption data from entire cities, regions, and countries. Energy consumption data on such a high level shows clear consumption trends. In this study, we apply the principles of VSTLF on individual households to model normal energy consumption behaviour. The energy consumption from individual households are more prone to differ from day to day. For example, simply arriving home a couple of hours late from work will alter the energy consumption behaviour for the majority of an evening.

#### 7.2.4 Related Work

Detecting abnormal energy consumption behaviour has gained more interest with the addition of smart meters. Smart meters allow both residents to be more aware of their own energy consumption and electricity providers to have a better estimate on how much energy is needed in an area. Zhang et al. [11] analyses energy consumption data on a household level to identify days when the residents have gone on vacation. They compare the accuracy of regression, entropy, and clustering based anomaly detection, with the regression based providing the best results. Chou and Telaga [12] investigate the possibility of real time anomaly detection for smart meter energy consumption in an office building. Auto-Regressive Integrated Moving Average (ARIMA) and a hybrid NN and ARIMA model were used to model the normal energy consumption behaviour. The authors used the two-sigma rule to determine if the actual energy consumption was abnormal. Meaning, if the actual energy consumption was more than 2 standard deviations away from the predicted energy consumption, it was labeled as abnormal.

Smart homes allow for a vast amount of different sources to collect data from. Research in the Ambient Assisted Living (AAL) area investigate the use of ambient sensors and actuators to determine if a resident is behaving out of the ordinary. Streaming data from the sensors and actuators can be used

directly to determine the residents normal behaviour and if any deviation has occurred [13]. Another possibility is to take the sensor data and map it into activities using Activity Recognition (AR). Identified activities are then used to determine how well they perform activities of daily living (ADLs). Sprint et al. [14] takes activities identified by AR and places them in time windows. These windows are compared to previous weeks' time windows to determine if a significant change has happened. The main drawback in these approaches is that smart homes are not that common yet. Installing sensors could also be seen as intrusive by the resident, and deploying this in a large scale takes a lot of effort.

Alcala et al. [15] focus on using energy consumption to determine if the residents behaviour is deviating from the norm. The authors use Non-Intrusive Load Monitoring (NILM), which takes a household's aggregated energy consumption and disaggregates it into individual appliances. They divide a day into time windows, and defines the probability of the appliances being used in these time windows using the Dempster-Shaffer theory. To detect abnormal activities, they compare the current day's time window to the same time window on previous days and calculate the probability of the residents deviating from the normal behaviour. However, disaggregating energy consumption into individual appliances is a complex task. The authors made the assumption that the disaggregation has already been performed, using data that was already disaggregated.

Chalmers et al. [16] investigate the use of smart meters and smart plugs to detect abnormal behaviour. They propose a system that analyses smart meter data, in conjunction with health data, to determine if a resident is deviating from their normal behaviour. The authors also conduct a case study of an elderly resident with a smart meter and smart plugs. They show differences in energy consumption on different mornings, arguing that abnormal behaviour could be identified using smart meter data. In line with the work of Chalmers et al., we have collected data and have started an initial study to investigate the possibility to detect abnormal behaviour using energy consumption data.

## 7.3 Data Analysis

We use energy consumption data measured at 1 minute intervals from 17 different households in Sweden. Initially, we received data from 20 households,

## 7. DETECTION OF RESIDENTS' ABNORMAL BEHAVIOUR BY ANALYSING ENERGY CONSUMPTION OF INDIVIDUAL HOUSEHOLDS

---

but 3 of them were discarded due to missing values. The data collection time spans from the 7th of February 2017 to the 1st of April 2017 and was provided by a Swedish electricity provider. In total for the 17 households, there are 1,311,720 measurements with 664 missing values (0.05%) remedied by applying linear interpolation.

Each household's average energy consumption and standard deviation is presented in Table 7.1. The households average energy consumption range from 0.0186 kWh to 0.0588 kWh per minute with standard deviations from 0.0076 kWh to 0.0322 kWh.

Table 7.1: Average energy consumption per minute and standard deviation for each household.

House	Mean (kWh)	STD (kWh)
A	0.031135	0.016534
B	0.018647	0.017853
C	0.035664	0.021904
D	0.030324	0.012230
E	0.036505	0.016897
F	0.056365	0.020543
G	0.019731	0.010121
H	0.051781	0.030274
I	0.058827	0.026903
J	0.037286	0.032240
K	0.029254	0.027309
L	0.027475	0.019119
M	0.038118	0.022269
N	0.036136	0.025799
O	0.038120	0.028139
P	0.043946	0.018218
Q	0.009349	0.007618

Two weeks of energy consumption for four households are shown in Figure 7.1, where the differences between the households' energy consumption are easier to identify. We can observe that the households are distinctive, both in terms of the *amount* of consumed energy and *how* the energy is consumed. For example, house D has a dense usage pattern where the energy consumption is focused around 0.03 kWh per minute. On the other hand, house K consistently has a low consumption during mornings and high peaks during late afternoons and evenings.

In general, it is not easy to identify a daily or weekly pattern for the households due to frequent variations in the data. This makes the identifi-

cation of an energy consumption trend even harder. Therefore, we create three additional data sets where we decrease the frequency and aggregate the energy consumption. The additional data sets have the intervals 10, 30, and 60 minutes.

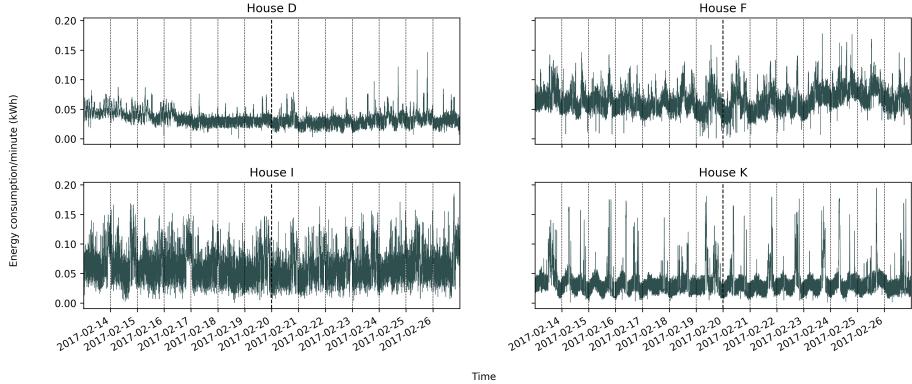


Figure 7.1: *Energy consumption of four different households from February 13-27 2017.*

Figure 7.2 shows the energy consumption of house K in the four data sets over two weeks (February 13-27, 2017). Comparing the different data sets, we can observe that with a frequency decrease it is easier to identify a daily pattern of energy consumption. With a larger time window between measurements we allow for more flexibility of the household residents. This yields a larger margin of error for how the residents use their appliances. In Figure 7.2 we see that peaks and slumps of consumption occur at roughly similar times each day. For example, February 22 and 23 are nearly identical, only differing in the height of the peak during the evening.

As described in Section 7.2.3, identification of daily and weekly trends is important for accurate energy consumption prediction. This section shows that there are energy consumption trends present in the data, albeit not that clear for all households.

## 7.4 Approach

In this section we present our approach to detect abnormal behaviour. First, we decide on what features we will include in our different feature sets. We

## 7. DETECTION OF RESIDENTS' ABNORMAL BEHAVIOUR BY ANALYSING ENERGY CONSUMPTION OF INDIVIDUAL HOUSEHOLDS

---

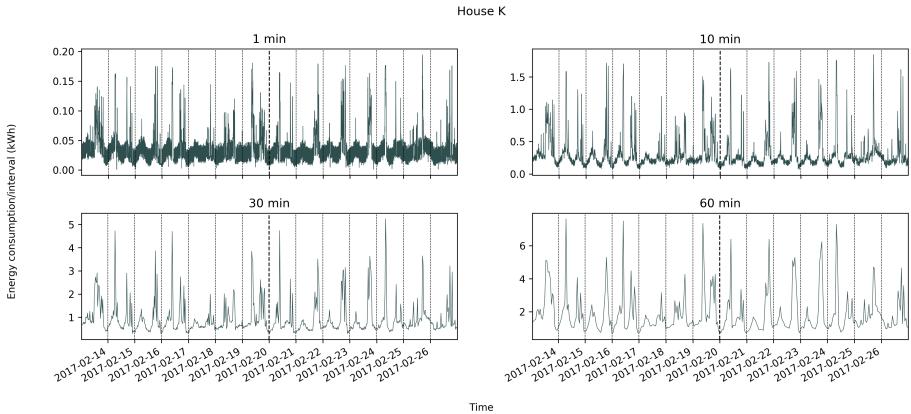


Figure 7.2: Energy consumption of household K in all four data sets from February 13-27 2017.

then choose three prediction algorithms to model normal behaviour. Finally, we describe our approach to detect anomalies.

### 7.4.1 Feature Selection

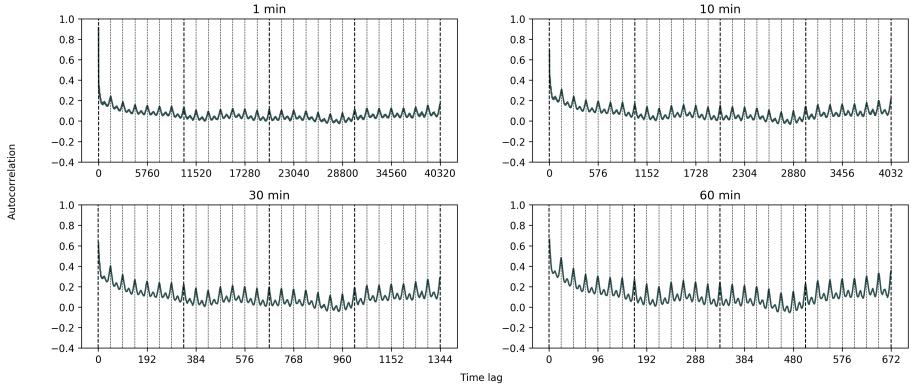


Figure 7.3: Average autocorrelation of all households for each data set up to four weeks back in time. Each vertical line represents a 24 hour time span.

As described in Section 7.3, we identified trends of energy consumption for our different households. Giving a good indication that using previous energy consumptions to predict future energy consumptions is a reasonable approach. This is in line with the literature of VSTLF where previous energy consumptions are extensively used as features [9, 17].

To decide which features to choose, we calculate the autocorrelation coefficient ( $r_k$ ) for all four data sets. The  $r_k$  coefficient gives a value of how much the energy consumption at time  $t$  correlates to the energy consumption  $k$  lags away. A lag in this context stands for the number of time periods between the measurements. For example, in the one minute data set one lag away would be one minute, two lags two minutes and so on. The autocorrelation is calculated as follows:

$$r_k = \frac{\sum_{t=k+1}^n (X_t - \bar{X})(X_{t-k} - \bar{X})}{\sum_{t=k+1}^n (X_t - \bar{X})^2} \quad (7.1)$$

where  $X_t$  is the energy consumption at time  $t$ ,  $\bar{X}$  is the average energy consumption, and  $X_{t-k}$  is the energy consumption at  $k$  lags away from  $t$ . Values range from -1 to 1, where a value closer to either -1 or 1 shows a strong negative or a positive correlation respectively. Values closer to 0 indicate less or no correlation between the measurements.

In Figure 7.3 we plot the average household autocorrelation for all data sets. We can observe that with lower measurement frequencies, we have a clearer correlation between measurements. Comparing 10, 30, and 60 minutes data sets to the 1 minute data set, the peaks in the graph are more distinctive. Giving an indication that it is easier to discover a consumption trend of individual households with a lower measurement frequency.

The strongest correlation in all data sets is the energy consumption one lag before the current. This drops off quickly and starts to enter a daily pattern of correlation. Each data set has a recurring trend of correlation at exact days before the current energy consumption. For example, in the 60 minute data set we have peaks of correlation at lag 24, 48, 72, and so on. These peaks of correlation indicate that using the energy consumption measurements at the same time from days before are suitable features.

To show the difference in correlation between the households, we plot the autocorrelation for two different households in Figure 7.4. We can observe a clear difference in how the energy is consumed between them. Household L follows the daily trend of energy consumption we have emphasized on. On the other hand, household B does not follow the daily trend but instead has a generally higher correlation for the whole week before the current energy consumption.

Based on the results from calculating the autocorrelation coefficient ( $r_k$ ) we design six different feature sets. The first two feature sets,  $F_a$  and  $F_b$ ,

## 7. DETECTION OF RESIDENTS' ABNORMAL BEHAVIOUR BY ANALYSING ENERGY CONSUMPTION OF INDIVIDUAL HOUSEHOLDS

---

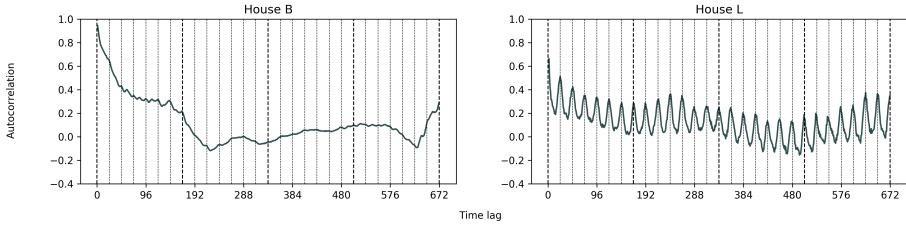


Figure 7.4: *Autocorrelation of two households. Each vertical line represents a 24 hour time span.*

both contain energy consumption measurements right before the load being predicted.  $F_a$  consists of 5 measurements right before the predicted at times  $t, \dots, t-4$ , and  $F_b$  contains 10 measurements at times  $t, \dots, t-9$ . Both feature sets consist of the measurements that have the highest correlation to the energy consumption being predicted. They are chosen as baseline feature sets, that only takes into account the energy consumption close to the predicted energy consumption. No daily or weekly trends are incorporated into these two feature sets.

In feature sets  $F_c$  and  $F_d$ , we incorporate weekly behaviour by adding energy consumption from the week previous to the prediction day as additional features.  $F_c$  is a super set of  $F_a$ , with the addition of energy consumption at times  $t+1, \dots, t-4$  from the same day one week before. Note that the time  $t+1$  was added from the previous week, as it has the highest correlation from that day to the value we want to predict.  $F_d$  has the same addition of features as  $F_c$ , only with more measurements, i.e. the energy consumption at times  $t+1, \dots, t-9$  from the previous week.

For feature set  $F_e$ , we aim to solely focus on daily behaviour in order to compare the effects of daily to weekly trends. We therefore take energy consumption from the lags right before the energy consumption that we want to predict and from the six consecutive days before. As we have more days to collect features from in this feature set, we only choose three measurements from each day. The features chosen were energy consumption at times  $t, t-1$  on the prediction day, and at times  $t+1, t, t-1$  on the six consecutive days before.

Finally, for  $F_f$  we continue with the weekly behaviour as we did in feature sets  $F_c$  and  $F_d$ .  $F_f$  is a super set of  $F_c$  with added energy consumption of two weeks before at times  $t+1, \dots, t-4$ . All feature sets are summarized

Table 7.2: Information about the feature sets extracted from our data sets.  $X_t$  stands for the energy consumption on the prediction day at time  $t$  and  $XD_t^n$  for the energy consumption  $n$  days before the prediction day at time  $t$ .

Feature set	Number of features	Variables
$F_a$	5	$X_t, \dots, X_{t-4}$
$F_b$	10	$X_t, \dots, X_{t-9}$
$F_c$	11	$F_a, XD_{t+1}^7, \dots, XD_{t-4}^7$
$F_d$	21	$F_b, XD_{t+1}^7, \dots, XD_{t-9}^7$
$F_e$	20	$X_t, X_{t-1}$ and $XD_{t+1}^n, XD_t^n, XD_{t-1}^n$ for $1 \leq n \leq 6$
$F_f$	17	$F_c, XD_{t+1}^{14}, \dots, XD_{t-4}^{14}$

in Table 7.2.

### 7.4.2 Prediction Algorithms

The first step to detecting abnormal behaviour is being able to model normal behaviour. Initially, we investigated how SVR, NNs, and LR perform in modelling the normal behaviour of all the 17 households. All three algorithms have shown good results for prediction in the VSTLF literature [5, 17]. However, initial experiments showed that NNs did not perform well with our data sets, therefore we discarded NNs from this study.

#### 7.4.2.1 Support Vector Regression

SVR is an extension of the Support Vector Machine (SVM) that allows for prediction of continuous values [18]. SVMs have been used to address a wide variety of classification problems, including image analysis [19] and text categorisation [20]. The SVM classifies instances by finding a hyperplane that separates two classes with the largest margin to any point within the training sets. A small subset of the data defines the decision boundary of the SVM, called support vectors. To create non-linear boundaries, the data can be projected into higher dimensions by using kernel functions. Both SVR and SVM rely on a small subset of the data to create their decision boundaries, making them tend to be resistant to overfitting the training data.

Initial tests showed that the linear and the radial basis kernel performed similarly well, which is why we choose both for our experiments. The linear kernel is configured with  $C = 250$  and  $\epsilon = 0.01$ , and the radial basis function

kernel with  $C = 500$ ,  $\epsilon = 0.005$ , and  $\gamma = 0.15$ . All parameters were chosen empirically.

#### 7.4.2.2 Linear Regression

LR is a statistical method used for forecasting. LR approximates the relationship between the independent variables and the dependent variable with a straight line, assuming there is a linear relationship between the two. The line that fits the data the best is chosen by minimizing the sum of squared errors, using ordinary least squares method.

#### 7.4.3 Anomaly Detection

We use the prediction algorithms presented in the previous subsection to detect anomalies. We calculate the residual between the actual energy consumption and the prediction models predicted energy consumption. The residual is then divided by the actual energy consumption, obtaining the relative distance between the two energy consumptions.

The relative distance,  $R$ , between the actual and predicted energy consumption is calculated as follows:

$$R = \frac{|F_t - A_t|}{A_t} \quad (7.2)$$

where  $F_t$  is the predicted energy consumption and  $A_t$  is the actual energy consumption at time  $t$ . If  $R > \alpha$ , where  $\alpha$  is a user defined parameter acting as a threshold, the energy consumption is considered abnormal.

### 7.5 Experiments

We divide our experiments into two parts;  $A$ : modelling normal behaviour, and  $B$ : identifying abnormal behaviour. In the first part, we determine the accuracy of the prediction algorithms when modelling the normal behaviour of the 17 households. In the second part, we investigate how different combinations of prediction algorithms, feature sets, and data sets detect abnormal behaviour. We use the implementations of each prediction algorithm from the Scikit-learn module [21].

### 7.5.1 Modelling Normal Behaviour

We perform a 5-fold cross validation to determine the performance of the prediction algorithms when modelling the normal behaviour. We use Mean Absolute Percentage Error (MAPE) to evaluate the prediction models which gives the accuracy of the predictions in percentage. It is defined as follows:

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \times 100 \quad (7.3)$$

where  $A_t$  is the actual value at time  $t$ ,  $F_t$  is the predicted value at time  $t$ , and  $n$  is the number of measurements.

### 7.5.2 Identifying Abnormal Behaviour

We randomly select three households from our data set to insert abnormal behaviour, households D, I, and K. The two scenarios defined in Section 7.2.1 are inserted in the final week of data separately, creating a data set for each scenario. Scenario S1, *sundowning*, we insert by taking the peak of consumption during a randomly selected evening, in this case the 29th of March, and repeat it for 4 additional hours. The additional 4 hours are the ones considered to be the abnormal behaviour. For households D and I, we take the energy consumption between 19:00 and 21:00 and repeat it until 01:00 the next day. For household K we choose the consumption between 16:00 and 18:00 and repeat it until 22:00.

Likewise, we insert scenario S2 the same way on the same day. We take the energy consumption of household D between 02:00 and 04:00 and repeat it until 08:00. For households I and K we choose the consumption between 01:00 and 03:00 and repeat it until 07:00.

We train the prediction algorithms on all the data leading up to the final week, i.e. the data from the 7th of February to, and including, the 25th of April. The final weeks energy consumption is predicted and the method described in Section 7.4.3 is used to determine if the energy consumption is normal or not. We then evaluate the anomaly detection by; 1) the detection rate (often referred as recall), which is the ratio between correctly classified anomalies and the number of anomalies, and 2) the false alarm rate, the ratio between normal data points classified as anomalous and the number of normal data points.

## 7. DETECTION OF RESIDENTS' ABNORMAL BEHAVIOUR BY ANALYSING ENERGY CONSUMPTION OF INDIVIDUAL HOUSEHOLDS

---

Table 7.3: MAPE scores from 5 fold cross validation for the three prediction algorithms on the 1 minute data set.

Feature set	Linear Regression	SVR (Linear)	SVR (RBF)
$F_a$	17.19	29.66	21.17
$F_b$	<b>16.87</b>	<b>29.06</b>	<b>20.76</b>
$F_c$	17.73	30.56	22.38
$F_d$	17.36	29.84	21.64
$F_e$	17.93	31.52	23.10
$F_f$	18.17	30.17	22.77

Table 7.4: MAPE scores from 5 fold cross validation for the three prediction algorithms on the 10 minute data set.

Feature set	Linear Regression	SVR (Linear)	SVR (RBF)
$F_a$	38.08	34.10	31.48
$F_b$	<b>35.56</b>	<b>32.73</b>	<b>28.88</b>
$F_c$	39.47	35.57	34.24
$F_d$	37.10	34.14	37.72
$F_e$	39.88	35.89	40.58
$F_f$	43.50	39.02	51.32

## 7.6 Results and Analysis

### 7.6.1 Cross Validation Results

The results from cross validating the 1, 10, 30, and 60 minute data sets are presented in Tables 7.3, 7.4, 7.5, and 7.6 respectively.

Comparing the four data sets, we observe that the one minute data set is the most accurate in modelling the normal behaviour. The MAPE scores are more stable for all prediction algorithms compared to the other data sets, varying between 16.87-18.17% for LR, 29.06-31.52% for SVR with linear kernel, and 20.76-23.10% for SVR with the RBF kernel. The one minute data set contains more energy consumption observations compared to the other data sets, i.e. we have more observations to train the prediction algorithms with, making the result in line with expectations.

However, the second most accurate data set is the 60 minute data set, where we have the least amount of observations. In the 60 minute data set SVR with linear kernel has its best accuracy, with a MAPE score between 21.30-22.75%. LR has its second best prediction accuracy with a MAPE between 23.86-24.86%. The SVR with RBF kernel does however vary a lot in

Table 7.5: MAPE scores from 5 fold cross validation for the three prediction algorithms on the 30 minute data set.

Feature set	Linear Regression	SVR (Linear)	SVR (RBF)
$F_a$	30.02	27.26	<b>26.07</b>
$F_b$	<b>29.88</b>	<b>27.12</b>	34.59
$F_c$	30.86	27.85	42.08
$F_d$	30.80	27.84	48.39
$F_e$	31.61	28.48	50.87
$F_f$	34.29	32.24	57.31

Table 7.6: MAPE scores from 5 fold cross validation for the three prediction algorithms on the 60 minute data set.

Feature set	Linear Regression	SVR (Linear)	SVR (RBF)
$F_a$	24.26	21.75	<b>28.81</b>
$F_b$	<b>23.86</b>	<b>21.30</b>	44.20
$F_c$	24.35	22.05	42.33
$F_d$	24.23	21.79	37.36
$F_e$	24.86	22.74	40.51
$F_f$	24.17	22.25	39.82

this data set, with a MAPE between 28.81-42.33%, which is also its second best accuracy in the data sets.

The different feature sets do affect the accuracy of modelling the normal behaviour. Overall, feature set  $F_b$  proves to be the most accurate for all data sets and prediction models, except for SVR with RBF kernel on the 30 and 60 minute data sets. Feature sets  $F_e$  and  $F_f$  seem to be alternating which is producing the worst results, depending on the data sets and the prediction algorithms. For example, in the one minute data set  $F_f$  provides the lowest accuracy for LR, while  $F_e$  is the lowest for SVR with linear kernel in the same data set.

We observe that SVR with RBF kernel has a varying prediction accuracy on our data. The accuracy on the one minute data set is among the best of all different configurations. However in the other 3 data sets, the accuracy is not consistent throughout the different feature sets, e.g. in the 30 minute data set where it has a MAPE of 26% on feature set  $F_a$  and a MAPE of 57.31% on feature set  $F_f$ . Both LR and SVR with linear kernel have consistent results on the 1 and 60 minute data sets, varying only 1-2%. In the 10 and 30 minute data sets, there is a larger variance in accuracy for

Table 7.7: Detection results for SVR linear kernel and Linear Regression on the 1 minute data set. DR stands for detection rate, the rate of identified anomalies, and FR stands for false alarm rate, the rate of normal instances classified as abnormal.

Scenario	Feature set	Linear Regression		SVR (Linear)	
		DR (%)	FR (%)	DR (%)	FR (%)
S1	$F_a$	8.33	26.12	11.39	40.48
	$F_b$	8.33	24.62	<b>12.22</b>	39.93
	$F_c$	8.47	25.56	11.25	39.43
	$F_d$	8.47	<b>24.13</b>	11.11	38.24
	$F_e$	8.75	25.42	11.67	38.16
	$F_f$	<b>9.44</b>	25.92	10.00	<b>33.31</b>
S2	$F_a$	34.44	26.04	<b>51.11</b>	40.26
	$F_b$	31.39	24.54	48.33	39.70
	$F_c$	34.03	25.47	49.31	39.25
	$F_d$	29.72	<b>24.04</b>	47.08	38.05
	$F_e$	33.19	25.32	48.75	38.01
	$F_f$	<b>34.58</b>	25.77	44.72	<b>33.08</b>

both algorithms.

### 7.6.2 Abnormal Behaviour

Based on the results presented in the previous section, we exclude the 10 and 30 minute data sets and the SVR with RBF kernel. The accuracy is not enough to build an accurate model of the normal behaviour. The  $\alpha$  parameter was set to 0.22 for the 1 minute data set and 0.27 for the 60 minute data set. The parameter values were chosen based on the MAPE scores from the previous section. The results for detecting the abnormal behaviour in the 1 minute data set are presented in Table 7.7 and the 60 minute data set in Table 7.8.

In the one minute data set, both LR and SVR have a poor detection rate for scenario S1. LR detecting 8-9% and SVR detecting 10-12% of the abnormal instances. However, between the two prediction algorithms we see a larger difference in the amount of normal instances classified as anomalies. LR has a false alarm rate of 24.62-26.12% while SVR has between 33.31-40-48%. For scenario S2 in the same data set, both algorithms have a higher detection rate. LR varying between 29.72-34.58% and SVR between 44.72-51.11% depending on the feature set.

In the 60 minute data set, the SVR remains at a similar false alarm rate

Table 7.8: Detection results for SVR linear kernel and Linear Regression on the 60 minute data set. DR stands for detection rate, the rate of identified anomalies, and FR stands for false alarm rate, the rate of normal instances classified as abnormal.

Scenario	Feature set	Linear Regression		SVR (Linear)	
		DR (%)	FR (%)	DR (%)	FR (%)
S1	$F_a$	<b>58.33</b>	48.47	<b>66.67</b>	38.04
	$F_b$	<b>58.33</b>	46.01	<b>66.67</b>	35.38
	$F_c$	41.67	46.83	50.00	37.83
	$F_d$	41.67	44.99	41.67	34.97
	$F_e$	41.67	40.70	41.67	35.79
	$F_f$	41.67	<b>40.29</b>	41.67	<b>34.56</b>
S2	$F_a$	<b>66.67</b>	48.26	16.67	38.24
	$F_b$	58.33	45.81	33.33	35.38
	$F_c$	<b>66.67</b>	46.63	33.33	37.83
	$F_d$	58.33	44.38	<b>41.67</b>	35.17
	$F_e$	<b>66.67</b>	40.49	25.00	35.79
	$F_f$	50.00	<b>40.29</b>	<b>41.67</b>	<b>33.54</b>

for both scenarios, differentiating 0-4% between the different feature sets. However, we can observe a clear increase of false alarm rates for LR, varying between 40.29-48.47% for scenario S1 and 40.29-48.26% for scenario S2. The detection rate for scenario S1 is increased substantially for both algorithms in the 60 minute data set. For scenario S2, LR also has an increased detection rate while SVR decreases.

Generally in both data sets, feature set  $F_f$  provides the least amount of false alarms for both algorithms. The feature set achieving the highest detection rate varies more between the different configurations, but overall  $F_a$  achieves the highest detection rate.

## 7.7 Discussions

Results from the first part of the experiments indicate that applying VSTLF on individual household's energy consumption to model normal behaviour is a promising approach. We achieve a prediction accuracy that is satisfactory and allow for detection of abnormal energy consumption behaviour. For further studies, we assume that setting parameters for the prediction algorithms on a household level, or clusters of households, would further improve the prediction accuracy.

The second part of the experiment demonstrate the possibility to detect

anomalous points of energy consumption. We assume that putting the point anomalies into context would allow us to draw more conclusions about the residents behaviour, especially with regards to health concerns. For example, a resident arriving at home one hour later than normal but follows the normal behaviour for the remainder of the evening. This may be flagged as abnormal behaviour, even though it is normal.

Initially, we expected the feature sets  $F_e$  and  $F_f$  to generate higher prediction accuracy and detection rate compared to the others. These feature sets capture more of the weekly and daily energy consumption behaviours. Instead,  $F_b$ , the feature set whose features are the energy consumptions right before the predicted, achieved the best prediction accuracy when modelling the normal behaviour. While  $F_f$  did provide the lowest false alarm rate for both data sets, the prediction accuracy when modelling the normal behaviour was generally the worst.

One possible reason to why  $F_e$  and  $F_f$  did not perform as expected is that both  $F_e$  and  $F_f$  have less data compared to  $F_b$ , in particular.  $F_e$  extracts features from the six previous days, causing it to have six days less of data instances compared to  $F_a$  and  $F_b$ . Likewise,  $F_f$  has features up to two weeks before. Having only seven weeks of data, this could be a reason why the performances of  $F_e$  and  $F_f$  are not in line with expectations.

The 1 minute and 60 minute data sets allowed for the best prediction accuracy for the prediction algorithms. LR had the best prediction accuracy of the entire experiment on the 1 minute data sets, and SVR had its best MAPE score on the 60 minute data set. We expected that the 60 minute data set would allow for better detection rates, which the results generally support. However, the false alarm rates for LR increases in the 60 minute data set compared to the 1 minute data set. The SVR did remain at similar levels for both data sets in terms of false alarm rate, but the detection rate was worse when detecting scenario S2 compared to the 1 minute data set.

## 7.8 Conclusions and Future Work

The aim of this paper is to conduct an initial study on using energy consumption data gathered from smart meters to detect abnormal behaviour of residents. We apply the concepts of VSTLF to build a model of normal behaviour and predict future energy consumption. The results from the

cross validation indicate that it is a promising approach to model the normal behaviour of the households.

Two scenarios indicating abnormal behaviour were defined and inserted into the data, and used for evaluating the prediction algorithms' ability to detect abnormal behaviour. Preliminary results indicate that it is possible to detect abnormal behaviour, but requires further study.

We have identified a number of directions for future work. First and foremost, we will collect data from a pilot study of elderlies in collaboration with our local municipality and eldercare. We will also investigate the addition of water consumption and weather data to further improve accuracy of modelling normal behaviour. We also plan to investigate the use of contextual anomalies instead of point anomalies to be able to draw more conclusions about the residents behaviour.

## Acknowledgements

We would like to thank IIOX and EON for their participation in this study and for providing the energy consumption data. This work is part of the research project “Scalable resource-efficient systems for big data analytics” funded by the Knowledge Foundation (grant: 20140032) in Sweden.

## References

- [1] U. Nations. “World population ageing: 1950-2050”. In: *New York: Department of Economic and Social Affairs* (2002).
- [2] X. Zhang, T. Kato, and T. Matsuyama. “Learning a context-aware personal model of appliance usage patterns in smart home”. In: *Innovative Smart Grid Technologies-Asia (ISGT Asia), 2014 IEEE*. IEEE. 2014, pp. 73–78.
- [3] L. K. Evans. “Sundown syndrome in institutionalized elderly”. In: *Journal of the American Geriatrics Society* 35.2 (1987), pp. 101–108.
- [4] V. Chandola, A. Banerjee, and V. Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

- [5] A. Setiawan, I. Koprinska, and V. G. Agelidis. "Very short-term electricity load demand forecasting using support vector regression". In: *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE. 2009, pp. 2888–2894.
- [6] T. Hong, M. Gui, M. E. Baran, and H. L. Willis. "Modeling and forecasting hourly electric load by multiple linear regression with interactions". In: *Power and Energy Society General Meeting, 2010 IEEE*. IEEE. 2010, pp. 1–8.
- [7] J. W. Taylor. "An evaluation of methods for very short-term load forecasting using minute-by-minute British data". In: *International Journal of Forecasting* 24.4 (2008), pp. 645–658.
- [8] H. S. Hippert, C. E. Pedreira, and R. C. Souza. "Neural networks for short-term load forecasting: A review and evaluation". In: *IEEE Transactions on power systems* 16.1 (2001), pp. 44–55.
- [9] I. Koprinska, M. Rana, and V. G. Agelidis. "Correlation and instance based feature selection for electricity load forecasting". In: *Knowledge-Based Systems* 82 (2015), pp. 29–40.
- [10] I. Drezga and S. Rahman. "Input variable selection for ANN-based short-term load forecasting". In: *IEEE Transactions on Power Systems* 13.4 (1998), pp. 1238–1244.
- [11] Y. Zhang, W. Chen, and J. Black. "Anomaly detection in premise energy consumption data". In: *Power and energy society general meeting, 2011 ieee*. IEEE. 2011, pp. 1–8.
- [12] J.-S. Chou and A. S. Telaga. "Real-time detection of anomalous power consumption". In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 400–411.
- [13] T. Rodner and L. Litz. "Data-driven generation of rule-based behavior models for an ambient assisted living system". In: *Consumer Electronics, Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on*. IEEE. 2013, pp. 35–38.
- [14] G. Sprint, D. Cook, R. Fritz, and M. Schmitter-Edgecombe. "Detecting Health and Behavior Change by Analyzing Smart Home Sensor Data". In: *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–3.

- [15] J. M. Alcalá, J. Ureña, Á. Hernández, and D. Gualda. “Assessing Human Activity in Elderly People Using Non-Intrusive Load Monitoring”. In: *Sensors* 17.2 (2017), p. 351.
- [16] C. Chalmers, W. Hurst, M. Mackay, and P. Fergus. “Smart Health Monitoring Using the Advance Metering Infrastructure”. In: *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2297–2302.
- [17] W. Charytoniuk and M.-S. Chen. “Very short-term load forecasting using artificial neural networks”. In: *IEEE transactions on Power Systems* 15.1 (2000), pp. 263–268.
- [18] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. “Support vector regression machines”. In: *Advances in neural information processing systems*. 1997, pp. 155–161.
- [19] O. Chapelle, P. Haffner, and V. N. Vapnik. “Support vector machines for histogram-based image classification”. In: *IEEE transactions on Neural Networks* 10.5 (1999), pp. 1055–1064.
- [20] T. Joachims. “Text categorization with support vector machines: Learning with many relevant features”. In: *Machine learning: ECML-98* (1998), pp. 137–142.
- [21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.



# Profiling of Household Residents' Electricity Consumption Behavior using Clustering Analysis

*Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz*

*In: International Conference on Computational Science, June, 2019, pp. 779-786, Springer, Faro, Portugal.*

## Abstract

In this study we apply clustering techniques for analyzing and understanding households' electricity consumption data. The knowledge extracted by this analysis is used to create a model of normal electricity consumption behavior for each particular household. Initially, the household's electricity consumption data are partitioned into a number of clusters with similar daily electricity consumption profiles. The centroids of the generated clusters can be considered as representative signatures of a household's electricity consumption behavior. The proposed approach is evaluated by conducting a number of experiments on electricity consumption data of ten selected households. The obtained results show that the proposed approach is suitable for data organizing and understanding, and can be applied for modeling electricity consumption behavior on a household level.

## 8.1 Introduction

The world's population is getting older. By 2050, projections state that the number of individuals over 60 will be around 2.1 billion. Keeping individuals in their own homes, and delaying their entrance to the health and elderly care systems, can help to offload costs and work from the already strained health care systems. Likewise, the elderly population often want to keep

## 8. PROFILING OF HOUSEHOLD RESIDENTS' ELECTRICITY CONSUMPTION BEHAVIOR USING CLUSTERING ANALYSIS

---

living independently at home, but they also want a sense of safety without any intrusion in their lives [1]. Remote monitoring, and assistance, is one way to provide the safety of the residents. Traditionally, remote monitoring has been performed with video surveillance. However, with the introduction of smart homes, i.e. houses with built in sensors and actuators, Ambient Assisted Living (AAL) has emerged and allows for monitoring and assistance without the use of cameras and with less intrusion of the residents' privacy.

With the adoption of smart meters in the electrical power grids, we have the opportunity to collect high resolution electricity consumption data remotely on a household level. This type of data can be used to get insight into the residents' habits and activities, with low impact and intrusion of the residents' privacy. We may detect abnormalities and changes of residents' behavior through analyzing their daily household electricity consumption. For example, dementia, and other neurodegenerative diseases, cause changes in the behavior of the individual in different ways, e.g., they can provoke insomnia, apathy, restlessness etc. [2]. We believe that changes like these, in the individual's daily behavior, can be caught by his/her electricity consumption activities.

Most current research related to household electricity consumption has mainly revolved around creating consumer profiles by clustering households together [3] and comparing different households to determine and predict abnormal consumption patterns [4]. But, there has been some research as well on household electricity consumption. For example, Zhang et al. [5] analyze energy consumption data on a household level to identify days when the residents have gone on vacation. Further, we have previously studied the use of prediction models for electricity consumption behavior [6].

In this paper, we present and evaluate a cluster analysis approach for organizing, understanding, and modeling household electricity consumption data, a continuation of our work in [7]. Our aim is to study the possibility of using the knowledge discovered by such analysis for creating consumption behavior signatures on a household level. The long-term goal is to investigate whether the created signatures can be used for identifying abnormal behavior in daily life and apply this outlier detection model in health care applications, e.g., for monitoring early stages of dementia or other neurodegenerative diseases. The developed consumption signatures can be considered as predefined activities and can be used for detecting abnormal consumption patterns

in order to notify the environment (relatives and health care professionals) if early signs of dementia occurs repeatedly at home.

## 8.2 Clustering Analysis Approach

### 8.2.1 Data Pre-Processing

The electricity consumption data collected in this study is gathered with a one-minute resolution and is measured in kWh (kilowatt hours). To be able to determine and profile a behavior of the household, we divide the time series into 24 hour profiles. This is an intuitive division of the data, as it allows us to capture a daily behavior which we then can analyze and use to model a routine daily behavior.

We set a maximum limit of 10% of the entire day or 20 consecutive minutes of missing data to remove that day from the data set. For the remaining profiles, we impute missing values by using linear interpolation. We then aggregate the electricity consumption data into a one hour resolution due to that resolution being more common for today's smart meters. Finally, we standardize the time series profiles using z-standardization, or Z-score, because we are more interested in the general shapes of the time series and not the actual amplitudes.

### 8.2.2 Clustering Algorithms & Validation Measures

Three partitioning algorithms are commonly used for data analysis to divide the data objects into  $k$  disjoint clusters [8]:  $k$ -means,  $k$ -medians, and  $k$ -medoids clustering. The three partitioning methods differ in how the cluster center is defined. In  $k$ -means clustering, the cluster center is defined as the mean data vector averaged over all objects in the cluster. In  $k$ -medians, the median is calculated for each dimension in the data vector to create the centroid. Finally, in  $k$ -medoids clustering, which is a robust version of the  $k$ -means, the cluster center is defined as the object with the smallest sum of distances to all other objects in the cluster, i.e., the most centrally located point in a given cluster. We have used  $k$ -medoids, since having an actual consumption profile as the cluster's centroid (medoid) is more representative of the consumption behavior compared to creating a synthetic centroid.

There are two major categories in which we can divide cluster validation measure to: *external* and *internal*. External measures are used when you

have prior knowledge of the data and validate according to the ground truth and internal measures validate based on the data and clusters themselves [9]. We use three internal validation measures for analyzing the data and to select the optimal clustering scheme. We have selected one validation measure for assessing compactness and separation - *Silhouette Index* [10], one for assessing connectedness - *Connectivity* [11], and one for assessing tightness and dealing with arbitrary shaped clusters - *IC-av* [12].

### 8.2.3 Distance Measures

The simplest and most widely used way to measure the distance, or dissimilarity, between two data points in a  $n$ -dimensional space (in our case two time series) is to calculate the Euclidean Distance (ED). ED calculates the distance between the two time series by aligning the  $i$ th point of one time series with the  $i$ th point of the other. ED is fast but it is sensitive to outliers and it cannot identify similarities between two segments if they are shifted out of phase [13]. Therefore, we also investigate the use of Dynamic Time Warping (DTW).

DTW measures the dissimilarity of two time series in a similar way, but instead of strictly calculating point by point it allows for some elasticity. One point in one of the time series can be aligned against one or more points in the other [14], which allows the identification of similar shapes even though they are out of phase.

## 8.3 Experiments and Results

### 8.3.1 Data

We have gathered electricity consumption data from 9909 anonymous households, collected with a 1-minute interval. Initially, the household data have gone through the pre-processing stage, as explained in Section 8.2.1. Then we have selected the 10 households with the largest number of daily profiles. 3 of these 10 households contained a few clearly abnormal profiles which we excluded from further analysis. At the final stage, the 10 selected households contain between 345 and 353 daily profiles. We then selected 1 of the 10 studied households as the representative and we discuss and interpret the results obtained on its data for the rest of our study.

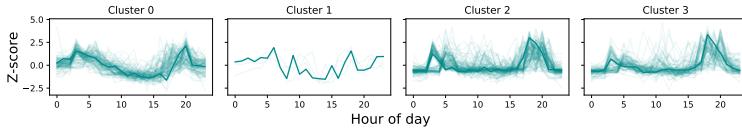


Figure 8.1: Clustering solution generated by  $k$ -medoids for  $k = 4$ , using DTW as a dissimilarity measure and supported to be the best by IC-av.

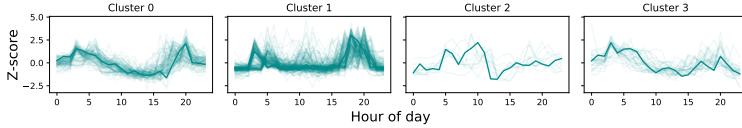


Figure 8.2: Clustering solution generated by  $k$ -medoids for  $k = 4$ , using DTW as a dissimilarity measure and supported to be the best by SI.

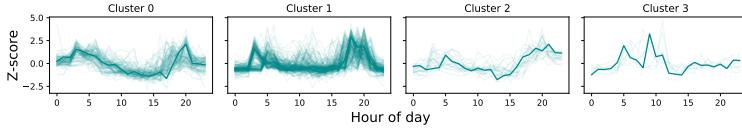


Figure 8.3: Clustering solution generated by  $k$ -medoids for  $k = 4$ , using DTW as a dissimilarity measure and supported to be the best by Connectivity.

## 8.3.2 Results and Analysis

### 8.3.2.1 Estimation of the Number of Clusters

We run the  $k$ -medoids clustering algorithm using the two distance metrics (ED and DTW) for all values of  $k$  between 2 and 9. The clustering algorithm is run 100 times for each  $k$  and with the cluster medoids randomly initialized. All clustering solutions are then evaluated using the cluster validation measures mentioned in Section 8.2.2. We then look upon each measure individually and in combination to determine which  $k$  is the appropriate. Based on the scores generated by the validation measures, we decide upon  $k = 5$  for ED and  $k = 4$  for DTW. However, the scores we evaluated are the best ones generated from each individual measure, i.e., the scores are not necessarily generated from the same clustering solutions.

### 8.3.2.2 Clustering Analysis

In Figures 8.1, 8.2, and 8.3 we show the clustering solutions produced by  $k$ -medoids using DTW as a distance metric. All three validation measures

## 8. PROFILING OF HOUSEHOLD RESIDENTS' ELECTRICITY CONSUMPTION BEHAVIOR USING CLUSTERING ANALYSIS

---

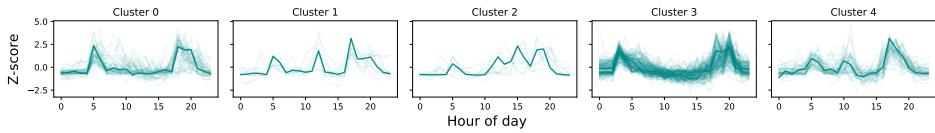


Figure 8.4: *Clustering solution generated by k-medoids for  $k = 5$ , using ED as a dissimilarity measure and supported to be the best by IC-av and Connectivity*

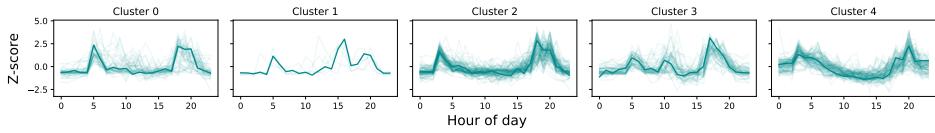


Figure 8.5: *Clustering solution generated by k-medoids for  $k = 5$ , using ED as a dissimilarity measure and supported to be the best by SI.*

support different clustering solutions. Therefore, we compare the three solutions and choose one of them that will be used to analyze the produced household consumption signatures. SI and Connectivity do however, share two clusters with the same signature and both of them contain mostly the same profiles. Both these solutions have two major and two smaller clusters. IC-av, on the other hand, generates a solution with three major clusters and only one small cluster. The solutions chosen by SI and Connectivity are fairly similar, with only a few profiles difference between them. Therefore, we have chosen the clustering solution supported by SI (Figure 8.2) as its smallest clusters contain more profiles compared to their counterparts in the solution preferred by Connectivity.

In Figures 8.4 and 8.5 we present the clustering solutions produced by ED. In this scenario, both IC-av and Connectivity support the same clustering solution, while SI has a solution of its own. However, the differences between the two solutions are not as distinct as in the case of DTW. We notice that they share the same signature for two of the clusters, and one additional cluster has a very similar signature. The solution promoted by SI does not have an equally as large cluster as the other solution has. In addition, in the solution proposed by IC-av and Connectivity we have two small clusters. In case of ED, we use the majority rule for choosing which clustering solution is the best and thereby it is the one proposed by IC-av and Connectivity (Figure 8.4).

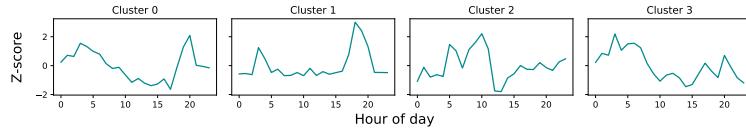


Figure 8.6: Electricity consumption signatures created from the cluster medoids from the chosen clustering solutions generated by  $k$ -medoids with  $k = 4$  (DTW).

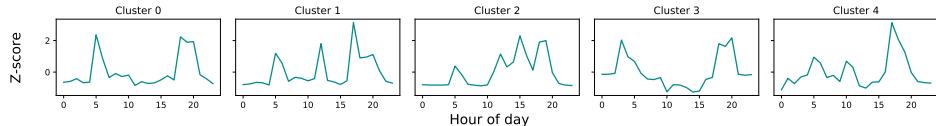


Figure 8.7: Electricity consumption signatures created from the cluster medoids from the chosen clustering solutions generated by  $k$ -medoids with  $k = 5$  (ED).

### 8.3.2.3 Consumption Behavior Signatures

The produced cluster centroids, which can be seen as the signatures of the electricity consumption habits of the household, are shown in Figures 8.6 and 8.7. We can see that the two different distance measures support different consumption signatures. For instance, DTW (Figure 8.6) focuses more on the general shape of the electricity consumption profiles, while ED (Figure 8.7) favours more the exact time of the days when the consumption peaks are happening. This supports our expectations, since DTW is an elastic measure that stretches the time series in the time axis to find an optimal alignment. Evidently, the different distance measures favour different electricity consumption profiles and logically, this will affect the intended analysis and built signatures. For example, we notice clear morning and evening consumption peaks recognized by Cluster 1 of the DTW solution (see Figure 8.6) and Clusters 0 and 3 of the ED signatures (see Figure 8.7), respectively. However, the additional consumption peak in the middle of the day seen in Cluster 1 of the ED solution is not clearly presented in any of the four signatures supported by DTW.

### 8.3.2.4 Context Based Signatures

In order to investigate further the electricity consumption behavior of households, we create context based signatures that represent how the weekdays are distributed between each cluster. These signatures are shown in Figures 8.8 and 8.9. They can be used to further improve and refine the

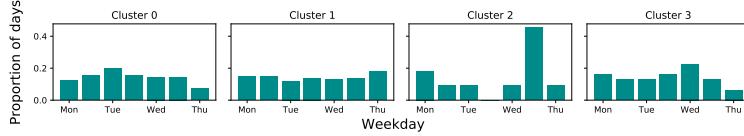


Figure 8.8: Context based signature of the weekdays from the clustering solution generated by  $k$ -medoids with  $k = 4$  (DTW).

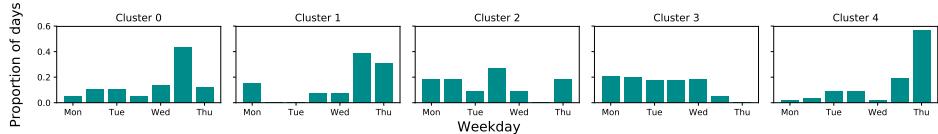


Figure 8.9: Context based signature of the weekdays from the clustering solution generated by  $k$ -medoids with  $k = 5$  (ED).

consumption behavior model and allow us to gain additional knowledge about the household’s behavior.

Comparing the distribution of weekdays produced by DTW (see Figure 8.8) and ED (see Figure 8.9), it is apparent that DTW divides the days more evenly than ED. The signatures of the three larger clusters, 0, 1, and 3, are fairly monotonic. ED, on the other hand, has a more distinct separation between working days and weekends. Clusters 0, 1, and 4 all contain more weekend days, while Cluster 3 has almost only working days. Cluster 2, which is the smallest cluster only containing 11 signatures, has a more diverse spread of its days. It is further interesting to notice that the signatures of Clusters 0 and 3 (see Figure 8.9) are very similar, which can be an indication that these could be merged into a single cluster. However, this is not strongly supported by the context presented in Figure 8.8, since the first cluster present a typical working day consumption behavior while the second one is more representative for the weekends.

### 8.3.3 Discussion

The produced electricity consumption signatures are representatives of the *current* electricity consumption behavior of the residents. To detect changes in the residents’ behavior, we can apply our approach on a new portion of data presenting the electricity consumption for the next time period. If new signatures are created through the clustering process, this might be a sign of a new behavior of the resident.

The proposed method can also be used to produce better prediction models. The generated clusters give a clearer distinction between normality and abnormality of the electricity consumption profiles. For example, in Figure 8.4 it is clear that clusters 0, 3, and 4, contain a majority of the electricity consumption profiles. Training the prediction models only on the contents of these clusters would give a more accurate model.

As mentioned before, it is beneficial to use some elasticity when comparing the electricity consumption profiles. Using DTW, we allow for changes in time for the individual electricity consumption profiles. However, we may cluster some signatures which probably should not be regarded as similar, e.g., if a resident is sick and stays in bed for a few extra hours and then performs his/her daily routines. We would like to identify this as a behavioral change, but DTW identifies this as normal. Introducing a time window for DTW to warp would remedy this.

## 8.4 Conclusions and Future Work

In this paper we propose a clustering analysis approach for profiling a households electricity consumption behavior. The proposed approach is evaluated on real electricity consumption data from 10 anonymous households. The results show that we can create electricity consumption signatures that model electricity consumption behaviors of the household residents. Further, we have found that Euclidean distance (ED) produce clusters that are more focused on the exact times of consumption peaks, Dynamic Time Warping (DTW) was better at identifying the shapes of the consumption peaks. We have also identified that ED has a clear distinction between working days and weekend days between the clusters, while DTW has a more monotonic distribution of days.

We are currently in the final stages of collecting both electricity and water consumption from a set of elderly residents with the help of our local elderly care system. The subjects will be continuously interviewed and monitored during the study to be able to accurately label the data, i.e. changes in their behavior. The collected data will be used for further evaluation and validation of the approach proposed in this study.

## References

- [1] W. L. Zagler, P. Panek, and M. Rauhala. "Ambient assisted living systems-the conflicts between technology, acceptance, ethics and privacy". In: *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fr Informatik. 2008.
- [2] M. S. Mega, J. L. Cummings, T. Fiorello, and J. Gornbein. "The spectrum of behavioral changes in Alzheimer's disease". In: *Neurology* 46.1 (1996), pp. 130–135.
- [3] T. Chen, A. Mutanen, P. Järventausta, and H. Koivisto. "Change detection of electric customer behavior based on AMR measurements". In: *PowerTech, 2015 IEEE Eindhoven*. IEEE. 2015, pp. 1–6.
- [4] C. Chalmers, W. Hurst, M. Mackay, and P. Fergus. "Profiling Users in the Smart Grid". In: *The Seventh International Conference on Emerging Networks and Systems Intelligence*. 2015.
- [5] Y. Zhang, W. Chen, and J. Black. "Anomaly detection in premise energy consumption data". In: *Power and energy society general meeting, 2011 ieee*. IEEE. 2011, pp. 1–8.
- [6] C. Nordahl, M. Persson, and H. Grahn. "Detection of Residents' Abnormal Behaviour by Analysing Energy Consumption of Individual Households". In: *1st Workshop on Aging, Rehabilitation, and Independent Assisted Living (ARIAL) 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2017, pp. 729–738.
- [7] C. Nordahl, V. Boeva, H. Grahn, and M. Netz. "Organizing, Visualizing and Understanding Households Electricity Consumption Data through Clustering Analysis". In: *Proceedings of the 2nd ARIAL Workshop, IJCAI 2018*. Stockholm, Sweden, June 2018, pp. 29–32.
- [8] J. MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *5th Berkeley Symp. on mathematical statistics and probability*. Vol. 1. 14. 1967, pp. 281–297.
- [9] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. "On clustering validation techniques". In: *Journal of intelligent information systems* 17.2-3 (2001), pp. 107–145.
- [10] P. J. Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

- [11] J. Handl, J. Knowles, and D. B. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.
- [12] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.
- [13] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. “Querying and mining of time series data: experimental comparison of representations and distance measures”. In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552.
- [14] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.



---

# Monitoring Household Electricity Consumption Behavior for Mining Changes

---

*Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz*

*In: 3rd International Workshop on AI for Aging, Rehabilitation and Independent Assisted Living (ARIAL), International Joint Conference on Artificial Intelligence (IJCAI), August, 2019, Macau, China.*

## Abstract

In this paper, we present an ongoing work on using a household electricity consumption behavior model for recognizing changes in sleep patterns. The work is inspired by recent studies in neuroscience revealing an association between dementia and sleep disorders and more particularly, supporting the hypothesis that insomnia may be a predictor for dementia in older adults. Our approach initially creates a clustering model of normal electricity consumption behavior of the household by using historical data. Then we build a new clustering model on a new set of electricity consumption data collected over a predefined time period and compare the existing model with the built new electricity consumption behavior model. If a discrepancy between the two clustering models is discovered a further analysis of the current electricity consumption behavior is conducted in order to investigate whether this discrepancy is associated with alterations in the resident's sleep patterns. The approach is studied and initially evaluated on electricity consumption data collected from a single randomly selected anonymous household. The obtained results show that our approach is robust to mining changes in the resident daily routines by monitoring and analyzing their electricity consumption behavior model.

## 9.1 Motivation and State of the Art

In this study, we develop a model that can be applied for monitoring household electricity consumption behavior and mining changes that can eventually be used to recognize shifts in routines and sleeping patterns of the resident, such as sleep disturbances.

Recent studies in neuroscience suggest a link between dementia and sleep disorders [1–3]. Insomnia and other sleep disturbances are common in patients with neurodegenerative disorders, such Alzheimer’s disease and other dementing disorders [2]. Changes in sleep of patients with Alzheimer’s disease are often observed on very early stage, e.g., a usual 20-minute daytime nap transforms into several hours per day. Sleep disorders can be an important diagnostic indication that foreruns development of Alzheimer’s disease pathological disorders. A systematic review published in [3] provides data supporting that insomnia may be a predictor for dementia in older adults.

Recently, sleep monitoring based on off-the-shelf mobile and wearable devices has emerged as a way to obtain information about one’s sleeping patterns [4], [5]. By taking advantage of diverse sensors, behaviors and routines associated with sleeping can be captured and modelled. For example, in [6] the bio-signals of a pool of Parkinson’s disease patients collected from ambient sensors have been analyzed to detect sleep disorders. What makes sensor monitoring particularly attractive is the non-invasive nature of the sensing compared for example, to traditional Polysomnography. Similarly, with the adoption of smart meters in the electrical power grids, we can now collect high resolution electricity consumption data remotely on a household level. This type of data can be used to get insight into the residents’ habits and activities, with low impact and intrusion of the residents’ privacy compared to sensor data. As it was discussed above, dementia and other neurodegenerative diseases can cause changes in the behavior of the individual by provoking insomnia, apathy, restlessness etc. We believe that such changes in an individual’s daily behavior, can be caught by their electricity consumption activities.

In our previous work, we have applied clustering techniques for analyzing and understanding households’ electricity consumption data [7]. The knowledge extracted from this analysis was then used to create a model of normal electricity consumption behavior for each particular household,

which was later described in [8].

In this work, we propose and investigate an approach that can be used to monitor such a model over time, detect changes and further analyze them on their meaningfulness. The proposed approach builds a new model on a new set of electricity consumption data collected over a predefined time period and then compares the existing historical model with the new household electricity consumption behavior model. If a discrepancy is discovered we perform a further analysis of the current electricity consumption behavior of the elderly habitant in order to investigate whether this discrepancy is associated with alterations in their daily routines. The proposed approach can also be applied to monitor and look for alterations in the sleep-wake cycle of elderly individuals who have already been diagnosed with dementia in order to avoid risk of falling, and identify the need for nursing home placement.

## 9.2 The Proposed Approach and Methods

In order to model the ordinary (normal) electricity consumption behavior of a household, we create  $k$  number of clusters with the use of  $k$ -medoids. The medoids of these clusters are then defined to model the normal electricity consumption behavior of the resident. We do not have information about the number of people living in the household, therefore we assume the resident lives alone. These signatures (the cluster medoids) are then used as a baseline to compare against for a new portion of data that arrives and also used as initial seeds for the clustering algorithm that is applied on the new data. The reason for using the existing cluster medoids as the initial seeds for the new data is the assumption that the new data will support the same electricity consumption behavior model as the old one. In addition, this enables the analysis and tracking of changes in the existing clusters and further detection of alterations in the electricity consumption behavior modes of the household. For example, we might trace cluster evolution through the detection of transitions, such as cluster shrinking, merging, splitting etc. This idea is schematically illustrated in Figure 9.1 by a bipartite graph, where the existing clusters (generated on historical data) and the newly generated clusters (used a new portion of data) are sets of left and right nodes and the arrows represents transition correlations between the clusters of two solutions.

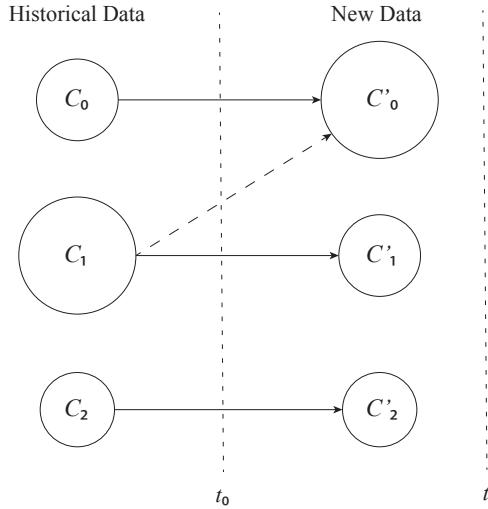


Figure 9.1: Schematic illustration of the proposed approach.

The new portion of data is also clustered with  $k$ -medoids, but as mentioned above the clustering is initialized by the medoids of the existing clustering solution. After the initial partitioning of the data into the clusters, the initial cluster medoids are removed from the data and new medoids are selected, then  $k$ -medoids iteratively continues to refine the clustering until it completes. We then compare the two clustering solutions (the existing and the new one) by calculating the Dynamic Time Warping (DTW) distance between their corresponding sets of medoids. If there is a discrepancy between the two clustering solutions, e.g., the calculated DTW is above the pre-defined threshold, a change in the electricity consumption behavior of the resident is identified. In such of case we further study how the existing clusters have evolved in the newly built clustering model in order to investigate whether this discrepancy is associated with alternations in the resident's sleep habits.

### 9.2.1 k-medoids

Three partitioning algorithms are commonly used for data analysis to divide the data objects into  $k$  disjoint clusters [9]:  $k$ -means,  $k$ -medians, and  $k$ -medoids clustering. The three partitioning methods differ in how the cluster center is defined. In  $k$ -means clustering, the cluster center is defined as the

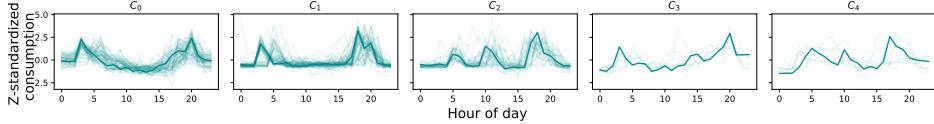


Figure 9.2: Cluster profiles of the clustering solution created by using data set  $A$

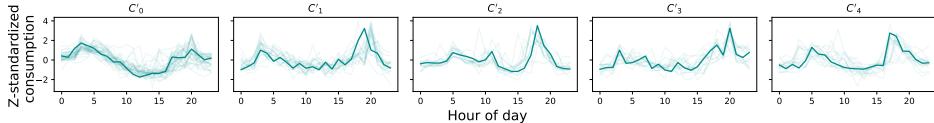


Figure 9.3: Cluster profiles of the clustering solution created by using data set  $B_0$

mean data vector averaged over all objects in the cluster. In  $k$ -medians, the median is calculated for each dimension in the data vector to create the centroid. Finally, in  $k$ -medoids clustering, which is a robust version of the  $k$ -means, the cluster center is defined as the object with the smallest sum of distances to all other objects in the cluster, i.e., the most centrally located point in a given cluster.

### 9.2.2 Dynamic Time Warping

The DTW alignment algorithm aims at aligning two sequences of feature vectors by warping the time axis iteratively until an optimal match (according to a suitable metrics) between the two sequences is found [10].

Let us formally explain how DTW works. Consider two matrices  $A = [a_1, \dots, a_n]$  and  $B = [b_1, \dots, b_m]$  with  $a_i$  ( $i = 1, \dots, n$ ) and  $b_j$  ( $j = 1, \dots, m$ ) column vectors of the same dimension. The two vector sequences  $[a_1, \dots, a_n]$  and  $[b_1, \dots, b_m]$  can be aligned against each other by arranging them on the sides of a grid, e.g. one on the top and the other on the left hand side. A distance measure, comparing the corresponding elements of the two sequences, can then be placed inside each cell. To find the best match or alignment between these two sequences one needs to find a path through the grid  $P = (1, 1), \dots, (i_s, j_s), \dots, (n, m)$ , ( $1 \leq i_s \leq n$  and  $1 \leq j_s \leq m$ ), which minimizes the total distance between  $A$  and  $B$ . Thus, the DTW distance between  $A$  and  $B$  can be defined as  $\text{dtw}(A, B) = \frac{1}{n+m} \min_P \left( \sum_{s=1}^k d(i_s, j_s) \right)$ .

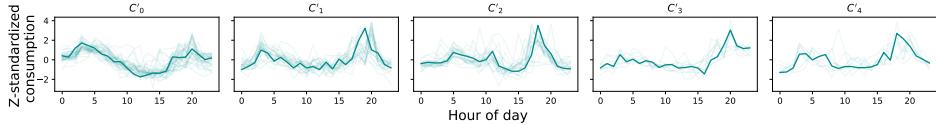


Figure 9.4: *Cluster profiles of the clustering solution created by using data set  $B_5$*

## 9.3 Initial Evaluation and Results

### 9.3.1 Data

In this study we use electricity consumption data collected from a single randomly selected anonymous household that has been collected with a 1-minute interval. We divide the data into two sets, one to represent the historical data that is used to model the ordinary electricity consumption behavior of the household (data set  $A$ ) and the other one to represent the new portion of data used to track changes in the existing household's behavior model (data set  $B$ ). Data set  $A$  contains 242 (70%) of the initial profiles and  $B$  the remaining 104 profiles (30%). For both sets we aggregate the electricity consumption data from 1-minute measures to 1-hour measures. Previous studies have indicated that using a too fine granularity makes it harder to identify patterns of consumption [11].

In order to study and evaluate the effectiveness of our approach, we introduce noise to data set  $B$  by altering a number of daily profiles. We insert 4 different amounts of noise to this data set, 5% ( $B_5$ ), 10% ( $B_{10}$ ), 15% ( $B_{15}$ ), and 20% ( $B_{20}$ ), and we retain one without any noise ( $B_0$ ). The noise insertion is performed by shifting one day  $x$  hours forward, where  $x$  is either (i) 12 hours or (ii) a random number of hours between 1 and 6. We choose these two different scenarios to mimic either (i) a complete change of the individual's sleeping pattern (12 hour shift) or (ii) a slighter deviation of the normal sleeping routine.

Then for all data sets we perform a z-standardization on all individual profiles, as we are more interested in the actual shapes during the studied period, i.e. how they consume, and not how much they consume. We argue that a normal behavior could look similar during the summer and winter, but the amplitudes may differ.

	$B_0$	$B_5$	$B_{10}$	$B_{15}$	$B_{20}$
$A$	0.0907	0.0955	0.0955	0.0889	0.0921

Table 9.1: The DTW distances between the existing cluster solution produced on data set  $A$  and the cluster solutions generated by the new data sets  $B_0$ ,  $B_5$ ,  $B_{10}$ ,  $B_{15}$  and  $B_{20}$ .

### 9.3.2 Experimental Setup

Initially, we apply  $k$ -medoids clustering on data set  $A$  in order to build the ordinary electricity consumption behavior model of the household. The optimal number of clusters ( $k$ ) is determined by initially studying and evaluating the used data by three different cluster validation indices: Silhouette Index [12], Connectivity [13], and Average Intra-Cluster distance [14]. In our experiments data set  $B$  represents the new portion of data collected over a predefined time period. In order to receive an insomnia diagnosis, patients must experience trouble falling or staying asleep for a period of one month or longer [15]. Therefore in our experiments the time axis is partitioned in monthly or longer intervals, i.e. the data set  $B$  covers a period longer than month.

We conduct five different experiments by considering five different new data sets:  $B_0$ ,  $B_5$ ,  $B_{10}$ ,  $B_{15}$ , and  $B_{20}$  (see Section 9.3.1). Each data set  $B_i$ , for  $i = 0, 5, 10, 15, 20$ , is clustered by using  $k$ -medoids initialized by the medoids of the clustering solution generated on data set  $A$ .

### 9.3.3 Results and Discussion

Table 9.1 lists the DTW distances between the existing cluster solution produced on data set  $A$  and the cluster solutions generated by the new data sets  $B_0$ ,  $B_5$ ,  $B_{10}$ ,  $B_{15}$  and  $B_{20}$ , respectively. One can notice the highest distances are recorded for data sets  $B_5$ ,  $B_{10}$  and  $B_{20}$ . Therefore, for the rest of this section we focus on studying the results produced by these three data sets and comparing them to the ones generated by data set  $B_0$ . The latter can be considered as a baseline for the new portion of data, since it has not been injected with any noise.

Table 9.2 and Table 9.3 present the DTW distances calculated between the cluster medoids of the existing clusters and the new clusters generated on data sets  $B_0$  and  $B_{20}$ , respectively. It is interesting to observe and trace

	$C'_0$	$C'_1$	$C'_2$	$C'_3$	$C'_4$
$C_0$	0.198	0.239	0.268	0.220	0.225
$C_1$	0.354	0.174	0.241	0.268	0.187
$C_2$	0.472	0.219	0.182	0.317	0.216
$C_3$	0.334	0.221	0.408	0.170	0.299
$C_4$	0.377	0.237	0.259	0.318	0.182

Table 9.2: DTW distances between the cluster medoids of the existing clusters ( $C_0, \dots, C_4$ ) and the new clusters ( $C'_0, \dots, C'_4$ ) generated on data set  $B_0$ .

	$C'_0$	$C'_1$	$C'_2$	$C'_3$	$C'_4$
$C_0$	0.198	0.239	0.335	0.252	0.249
$C_1$	0.354	0.174	0.182	0.256	0.218
$C_2$	0.472	0.219	0.164	0.304	0.293
$C_3$	0.334	0.221	0.364	0.199	0.228
$C_4$	0.377	0.237	0.213	0.281	0.186

Table 9.3: DTW distances between the cluster medoids of the existing clusters ( $C_0, \dots, C_4$ ) and the new clusters ( $C'_0, \dots, C'_4$ ) generated on data set  $B_{20}$ .

	$C'_0$	$C'_1$	$C'_2$	$C'_3$	$C'_4$
$B_0$	36	21	15	16	16
$B_5$	36	25	22	10	9
$B_{10}$	36	23	21	13	9
$B_{15}$	34	20	15	14	19
$B_{20}$	37	23	22	10	10

Table 9.4: Distribution of daily profiles to different clusters for all  $B$  data sets.

the evolution of clusters  $C_1$  and  $C_4$  in data set  $B_{20}$  compared to data set  $B_0$ . For example,  $C_1$  gets closer to  $C'_2$ , while  $C_4$  moves towards clusters  $C'_2$  and  $C'_3$ . These observations are also supported by the heatmaps plotted in Figure 9.5 and Figure 9.6, respectively. One can observe that the respective cells in the heatmap of  $B_{20}$  have changed their colour in comparison with the heatmap of the baseline data set  $B_0$ . In addition, as it can be seen in Table 9.4, cluster  $C'_4$  which is a transition of cluster  $C_4$  in the newly generated clustering solution gets shrunk in all three studied data sets ( $B_5$ ,  $B_{10}$  and  $B_{20}$ ), while logically cluster  $C'_2$  gets large.

We study further the above discussed observations in Figures 9.2, 9.3 and 9.4, which depict cluster profiles of the clustering solutions generated on data sets  $A$ ,  $B_0$ ,  $B_5$ , respectively.  $C_0$  and  $C_1$  are the biggest clusters and represent the electricity consumption behavior more typical for working days with clearly recognized morning and evening consumption peaks. Cluster  $C_2$  is also comparatively big and has an additional consumption peak in the middle of the day, i.e. it models behavior more typical for the weekends.

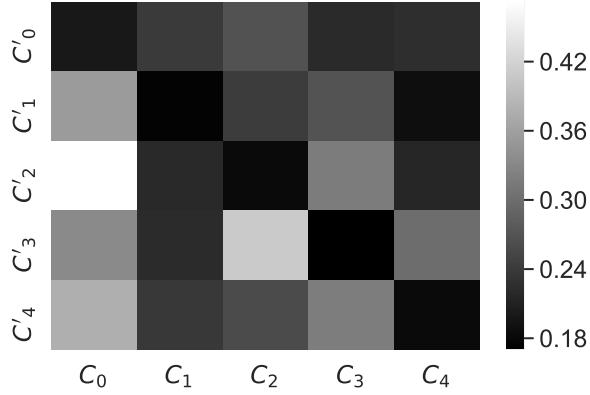


Figure 9.5: Heatmap for distances between the existing clusters and the clusters generated by  $B_0$ .

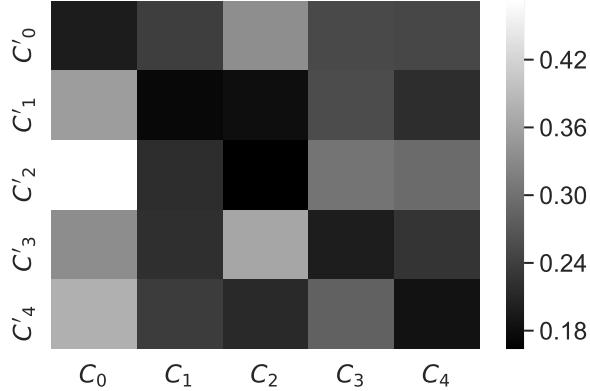


Figure 9.6: Heatmap for distances between the existing clusters and the clusters generated by  $B_{20}$ .

Clusters  $C_3$  and  $C_4$  are smaller and do not represent so clearly recognizable behavior. Now let us trace how these five clusters are evolved in the new data sets  $B_0$  and  $B_5$ , respectively. As we can see in Table 9.4 clusters  $C'_0$  and  $C'_1$  are also the biggest ones in both data sets ( $B_0$  and  $B_5$ ) and have very similar shapes to those of  $C_0$  and  $C_1$  (see Figures 9.3 and 9.4). However, cluster  $C'_2$  does not have a similar profile to that of its origin cluster  $C_2$ . It models behavior that shows electricity consumption activity too early in the morning and in addition, it expands in data set  $B_5$ . The transformation of cluster  $C_4$  in data sets  $B_0$  and  $B_5$  also has an effect on its shape. We

can observe that  $C'_4$  does not have an additional consumption peak in the middle of the day like  $C_4$ .

## 9.4 Conclusions and Future Work

In this paper, we have proposed an approach for monitoring household electricity consumption behavior and mining changes that can be used to recognize shifts in daily routines and sleeping patterns of the resident. The proposed approach uses clustering techniques for modelling and analyzing household's electricity consumption behavior. It has been initially evaluated on electricity consumption data collected from a single randomly selected anonymous household. The experimental results have shown that the approach is robust to recognizing alternations in the resident daily routines by monitoring and analyzing their electricity consumption behavior.

Our future plans are to pursue further evaluation of the proposed approach by involving healthcare experts and using data from other sources, e.g. ambient sensors, for its validation on richer data and real scenarios. In a long-term perspective, we are interested in applying the developed approach for recognizing early signs of dementia by monitoring and tracing changes in household electricity consumption behavior in addition to other data sources.

## References

- [1] A. Brzecka, J. Leszek, G. M. Ashraf, M. Ejma, M. F. Ávila-Rodriguez, N. S. Yarla, V. V. Tarasov, V. N. Chubarev, A. N. Samsonova, G. E. Barreto, and G. Aliev. "Sleep Disorders Associated With Alzheimer's Disease: A Perspective". In: *Frontiers in Neuroscience* 12 (2018), p. 330.
- [2] G. Cirpiani, C. Lucetti, S. Danti, and A. Nuti. "Sleep disturbances and dementia". In: *Psychogeriatrics, Japanese Psychogeriatric Society* 15 (2015), pp. 65–74.
- [3] K. M. de Almondes, M. V. Costa, L. F. Malloy-Diniz, and B. S. Diniz. "Insomnia and risk of dementia in older adults: Systematic review and meta-analysis". In: *Journal of Psychiatric Research* 77 (2016), pp. 109–115.

- 
- [4] A. V. Shelgikar, P. F. Anderson, and M. R. Stephens. “Sleep tracking, wearable technology, and opportunities for research and clinical care”. In: *Chest* 150.3 (2016), pp. 732–743.
  - [5] K. Ping-Ru, J. A. Kientz, E. K. Choe, M. Kay, C. A. Landis, and N. F. Watson. “Consumer Sleep Technologies: A Review of the Landscape”. In: *Journal of Clinical Sleep Medicine* 11.12 (2015), pp. 1455–1461.
  - [6] S. Buet. *Analysis of sleep-related symptoms of Parkinson's patients based on a system of ambient sensors*. Zurich, Switzerland: Master Thesis, ETHZ, Department of Information Technology and Electrical Engineering, 2017.
  - [7] C. Nordahl, V. Boeva, H. Grahn, and M. Netz. “Organizing, Visualizing and Understanding Households Electricity Consumption Data through Clustering Analysis”. In: *Proceedings of the 2nd ARIAL Workshop, IJCAI 2018*. Stockholm, Sweden, June 2018, pp. 29–32.
  - [8] C. Nordahl, V. Boeva, H. Grahn, and M. Netz. “Profiling of Household Residents' Electricity Consumption Behavior using Clustering Analysis”. In: *Proceedings of International Conference on Computational Science*. Faro, Algarve, Portugal: Springer, June 2019.
  - [9] J. MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *5th Berkeley Symp. on mathematical statistics and probability*. Vol. 1. 14. 1967, pp. 281–297.
  - [10] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.
  - [11] C. Nordahl, M. Persson, and H. Grahn. “Detection of Residents' Abnormal Behaviour by Analysing Energy Consumption of Individual Households”. In: *1st Workshop on Aging, Rehabilitation, and Independent Assisted Living (ARIAL) 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2017, pp. 729–738.
  - [12] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
  - [13] J. Handl, J. Knowles, and D. B. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.

- [14] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.
- [15] T. Roth. “Insomnia: definition, prevalence, etiology, and consequences”. In: *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine* 3.5 Suppl (2007), S7.

---

# Modeling Evolving User Behavior via Sequential Clustering

*Veselka Boeva and Christian Nordahl*

*In: 2nd International Workshop on Knowledge Discovery and User Modelling for Smart Cities (UMCit), European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), September, 2019, Würzburg, Germany.*

## Abstract

In this paper we address the problem of modeling the evolution of clusters over time by applying sequential clustering. We propose a sequential partitioning algorithm that can be applied for grouping distinct snapshots of streaming data so that a clustering model is built on each data snapshot. The algorithm is initialized by a clustering solution built on available historical data. Then a new clustering solution is generated on each data snapshot by applying a partitioning algorithm seeded with the centroids of the clustering model obtained at the previous time interval. At each step the algorithm also conducts model adapting operations in order to reflect the evolution in the clustering structure. In that way, it enables to deal with both incremental and dynamic aspects of modeling evolving behavior problems. In addition, the proposed approach is able to trace back evolution through the detection of clusters' transitions, such as splits and merges. We have illustrated and initially evaluated our ideas on household electricity consumption data. The results have shown that the proposed sequential clustering algorithm is robust to modeling evolving behavior by being enable to mine changes and update the model, respectively.

## 10.1 Introduction

The need for describing and understanding the behavior of a given phenomenon over time led to the emergence of new techniques and methods focused in temporal evolution of data and models [1–3]. Data mining techniques and methods that enable to monitor models and patterns over time, compare them, detect and describe changes, and quantify them on their interestingness are encompassed by the paradigm of change mining [4]. The two main challenges of this paradigm are to be able to adapt models to changes in data distribution but also to analyze and understand changes themselves.

Evolving clustering models are referred to incremental or dynamic clustering methods, because they can process data step-wise and update and evolve cluster partitions in incremental learning steps [5], [6]. Incremental (sequential) clustering methods process one data element at a time and maintain a good solution by either adding each new element to an existing cluster or placing it in a new singleton cluster while two existing clusters are merged into one [7], [8], [9]. Dynamic clustering is also a form of online/incremental unsupervised learning. However, it considers not only incrementality of the methods to build the clustering model, but also self-adaptation of the built model. In that way, incrementality deals with the problem of model re-training over time and memory constrains, while dynamic aspects (e.g., data behavior, clustering structure) of the model to be learned can be captured via adaptation of the current model. Lughofe proposes an interesting dynamic clustering algorithm which is also dedicated to incremental clustering of data streams and in addition, it is equipped with dynamic split-and-merge operations [6]. A similar approach defining a set of splitting and merging action conditions is introduced in [10]. Wang et al. also propose a split-merge-evolve algorithm for clustering data into  $k$  number of clusters [11]. However, a  $k$  cluster output is always provided by the algorithm, i.e. it is not sensitive to the evolution of the data. A split-merge evolutionary clustering algorithm which is robust to evolving scenarios is introduced in [12]. The algorithm is designed to update the existing clustering solutions based on the data characteristics of newly arriving data by either splitting or merging existing clusters. Notice that all these algorithms have the ability to optimize the clustering result in scenarios where new data samples may be added in to existing clusters.

In this paper, we propose a sequential (dynamic) partitioning algorithm that is robust to modeling the evolution of clusters over time. In comparison with the above discussed dynamic clustering algorithms it does not update existing clustering, but groups distinct portions (snapshot) of streaming data so that a clustering model is generated on each data portion. The algorithm initially produces a clustering solution on available historical data. A clustering model is generated on each new data snapshot by applying a partitioning algorithm initialized with the centroids of the clustering solution built on the previous data snapshot. In addition, model adapting operations are performed at each step of the algorithm in order to capture the clusters' evolution. Hence, it tackles both incremental and dynamic aspects of modeling evolving behavior problems. The algorithm also enables to trace back evolution through the identification of clusters' transitions such as splits and merges. We have studied and initially evaluated our algorithm on household electricity consumption data. The results have shown that it is robust to modeling evolving data behavior.

## 10.2 Modeling Evolving User Behavior via Sequential Clustering

### 10.2.1 Sequential Partitioning Algorithm

In this section, we formally describe the proposed sequential partitioning algorithm. The algorithm idea is schematically illustrated in Figure 10.1.

Assume that data sets  $D_0, D_1, \dots, D_n$  are distinct snapshots of data stream. Further let  $C = \{C_i | i = 0, 1, \dots, n\}$  be a set of clustering solutions (models), such that  $C_i$  has been built on a data set  $D_i$ . In addition, each clustering solution  $C_i$ , for  $i = 1, 2, \dots, n$ , is generated by applying a partitioning algorithm (see Section 10.2.2) on data set  $D_i$  initialized (seeded) with the centroids of the clustering model built on data set  $D_{i-1}$ . The algorithm is initialized by clustering  $C_0$  which is extracted from data set  $D_0$  (available historical data or the initial snapshot).

The basic operations conducted by our algorithm at each time window (on each data snapshot)  $i$  are explained below:

1. **Input:** Cluster centroids of partition  $C_{i-1}$  ( $i = 1, 2, \dots, n$ ).

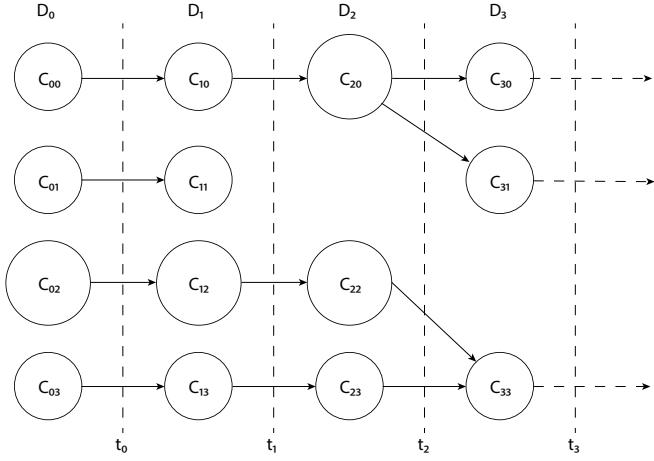


Figure 10.1: *Schematic illustration of the proposed sequential clustering approach*

2. **Clustering step:** Cluster data set  $D_i$  by seeding the partitioning algorithm with the centroids of  $C_{i-1}$ .
  - (a) Initial clustering of  $D_i$ .
  - (b) Check for empty initial clusters and adapt the partitioning respectively.
  - (c) Remove the seeded centroids and finalize the clustering by producing  $C_i$ .
3. **Adapting step:** For each cluster  $C_{ij} \in C_i$  do the following steps
  - (a) Calculate split condition for  $C_{ij}$ .
  - (b) If the split condition is satisfied then split  $C_{ij}$  into two clusters by applying 2-means clustering algorithm and update the list of centroids, respectively.
4. **Output:** Updated clustering partition and list of centroids used to initialize the clustering action that will be conducted on data set  $D_{i+1}$ .

Note that at step 2(b) above we check whether there are empty clusters after the initial clustering. If so, this means that the clustering structure is evolving, i.e. some clusters may stop existing while others are merged

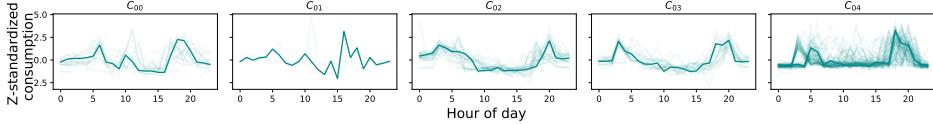


Figure 10.2: Initial clustering model produced on the historical data.

together. Evidently, the death and merge transitions are part of the clustering step. Therefore the split condition is only checked at step 3<sup>1</sup>. We can apply different split conditions. For example, the homogeneity of each cluster  $C_{ij}$  may be evaluated and if it is below a given threshold we will perform splitting. Another possibility is to apply the idea implemented by Lugofer [6] in his dynamic split-and-merge algorithm.

In order to trace back the clusters' evolution we can compare the sets of cluster centroids of each pair of partitioning solutions extracted from the corresponding neighborhood time intervals (e.g., see Figure 10.6). This comparison can be performed by applying some alignment technique, e.g., such as Dynamic Time Warping (DTW) algorithm explained in Section 10.2.3. For example, if we consider two consecutive clustering solutions  $C_{i-1}$  and  $C_i$  ( $i = 1, 2, \dots, n$ ), we can easily recognize two scenarios: (i) a centroid of  $C_{i-1}$  is aligned to two or more centroids of  $C_i$  then the corresponding cluster from  $C_{i-1}$  splits among the aligned ones from  $C_i$ ; (ii) a few centroids of  $C_{i-1}$  is aligned to a centroid of  $C_i$  then the corresponding clusters from  $C_{i-1}$  merge into the aligned cluster from  $C_i$ .

### 10.2.2 Partitioning Algorithms

Three partitioning algorithms are commonly used for data analysis to divide the data objects into  $k$  disjoint clusters [13]:  $k$ -means,  $k$ -medians, and  $k$ -medoids clustering. The three partitioning methods differ in how the cluster center is defined. In  $k$ -means clustering, the cluster center is defined as the mean data vector averaged over all objects in the cluster. In  $k$ -medians, the median is calculated for each dimension in the data vector to create the centroid. Finally, in  $k$ -medoids clustering, which is a robust version of the  $k$ -means, the cluster center is defined as the object with the smallest sum of distances to all other objects in the cluster, i.e., the most centrally located point in a given cluster.

<sup>1</sup> Step 3 is not implemented into the current version of our sequential clustering algorithm.

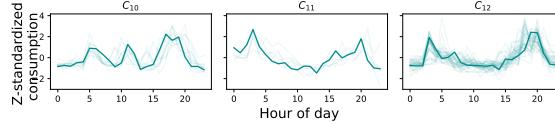


Figure 10.3: Clustering model produced on the first new data snapshot. Clusters  $C_{01}$  and  $C_{03}$  have been empty after the clustering step. Clusters  $C_{02}$  and  $C_{04}$  are transformed in clusters  $C_{11}$  and  $C_{12}$ , respectively.

### 10.2.3 Dynamic Time Warping Algorithm

The DTW alignment algorithm aims at aligning two sequences of feature vectors by warping the time axis iteratively until an optimal match (according to a suitable metrics) between the two sequences is found [14]. Let us consider two matrices  $A = [a_1, \dots, a_n]$  and  $B = [b_1, \dots, b_m]$  with  $a_i$  ( $i = 1, \dots, n$ ) and  $b_j$  ( $j = 1, \dots, m$ ) column vectors of the same dimension. The two vector sequences  $[a_1, \dots, a_n]$  and  $[b_1, \dots, b_m]$  can be aligned against each other by arranging them on the sides of a grid, e.g. one on the top and the other on the left hand side. A distance measure, comparing the corresponding elements of the two sequences, can then be placed inside each cell. To find the best match or alignment between these two sequences one needs to find a path through the grid  $P = (1, 1), \dots, (i_s, j_s), \dots, (n, m)$ , ( $1 \leq i_s \leq n$  and  $1 \leq j_s \leq m$ ), which minimizes the total distance between  $A$  and  $B$ .

## 10.3 Case Study: Modeling Household Electricity Consumption Behavior

### 10.3.1 Case Description

Suppose that a monitoring system for tracking changes in electricity consumption behavior at a household level is developed to be used for some healthcare application. For example, such a system can be used to monitor and look for alterations in the daily routines (sleep-wake cycle) of elderly individuals who have been diagnosed with a neurodegenerative disease. The system is supposed to build and maintain an electricity consumption behavior model for each monitored household. Initially, a model of normal electricity consumption behavior is created for each particular household by using historical data [15]. In order to monitor such a model over time it is necessary to build a new model on each new portion of electricity con-

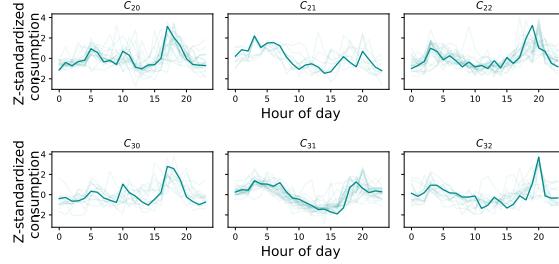


Figure 10.4: Clustering model produced on the second (above) and third (below) new data snapshots, respectively.

sumption data and then compare the current model with the new household electricity consumption behavior model. If changes are identified a further analysis of the current electricity consumption behavior model is performed in order to investigate whether these are associated with alterations in the resident's daily routines.

### 10.3.2 Data and Experiments

We use electricity consumption data collected from a single randomly selected anonymous household that has been collected with a 1-minute interval for a period of 14 months. During those 14 months, there were roughly 2 months worth of data that had not been collected, i.e. zero values which have been removed. We then aggregate the electricity consumption data into a one hour resolution from the one minute resolution.

We divide the data into four parts. The first 50% of the total data represent the historical data ( $D_0$ ), and the remaining data is evenly distributed into the other three data sets ( $D_1$ ,  $D_2$  and  $D_3$ ). In addition,  $D_2$  and  $D_3$  have their contents shifted to simulate a change in their behavior over time. 8% of the contents in  $D_2$  is randomly shifted 1 to 6 hours ahead, and 2% of the contents 12 hours ahead. Similarly,  $D_3$  has 16% of the data shifted 1 to 6 hours ahead and 4% 12 hours ahead. We choose these two scenarios to simulate both minor and drastic changes in the sleeping pattern of the resident.

In order to cluster the historical data, we run  $k$ -medoids 100 times using randomly initialized cluster medoids, for each  $k$  between 2 and 20. DTW is used as the dissimilarity measure and it is restricted to only allow for a maximum warp of two hours. This restriction is in place to allow for some

Table 10.1: *Distances between the clustering models generated on the first and second new data snapshots (left), and on the second and third new data snapshots (right), respectively.*

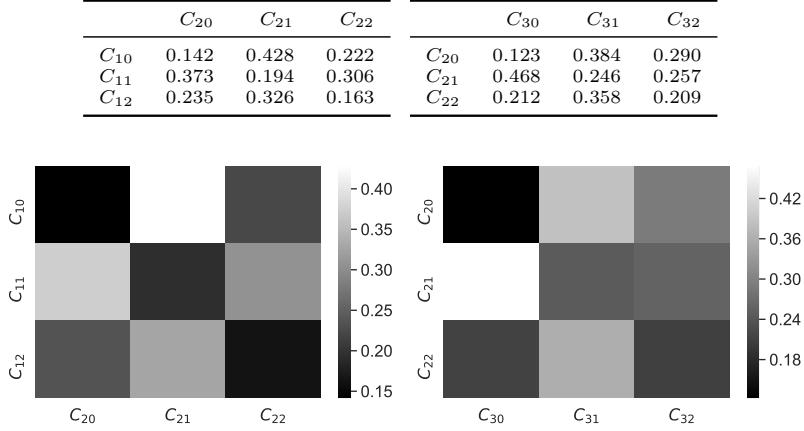


Figure 10.5: *Heatmaps for distances between the clustering models generated on the first and second new data snapshots (left), and on the second and third new data snapshots (right), respectively.*

minor alterations in the daily behavior while keeping major alterations in check. The produced clustering solutions are then evaluated using Silhouette Index [16], Connectivity [17], and Average Intra-Cluster distance [18]. The medoids from the best scoring clustering solution are then used as the initial seeds for the next snapshot of data, as explained in Section 10.2.1.

### 10.3.3 Results and Discussion

Figure 10.2 shows the initial clustering model generated on the historical data. As one can see the household electricity consumption behavior is modeled by five different behavior profiles (clusters).  $C_{03}$  and  $C_{04}$  are the biggest clusters and represent the electricity consumption behavior more typical for working days with clearly recognized morning and evening consumption peaks. Clusters  $C_{00}$  and  $C_{01}$  are smaller and have an additional consumption peak in the middle of the day, i.e. they model behavior more typical for the weekends. Cluster  $C_{02}$  is comparatively big and represents electricity consumption behavior typical for working days with a slightly later start.

Figure 10.3 depicts the clustering model derived from the first new data snapshot. As we can notice the electricity consumption behavior is

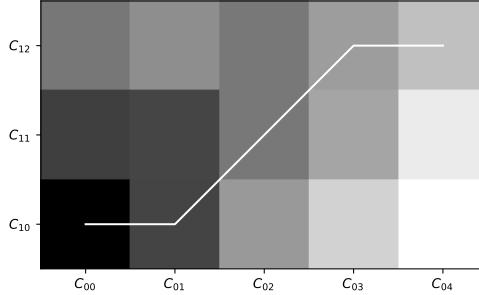


Figure 10.6: DTW alignment path between the clustering models generated on the historical and first new snapshot data, respectively.

modeled only by three clusters.  $C_{01}$  and  $C_{03}$  have been empty after the initial clustering step and their medoids are removed from the list of medoids. Clusters  $C_{02}$  and  $C_{04}$  are transformed into clusters  $C_{11}$  and  $C_{12}$ , respectively. It is interesting to observe that  $C_{12}$  is also very similar to cluster profile  $C_{03}$ . The latter observation is also supported by the DTW alignment between the medoids of the two clustering models given in Figure 10.6, where  $C_{03}$  and  $C_{04}$  are aligned to  $C_{12}$ , i.e. they are merged into one cluster. This is also the case for  $C_{00}$  and  $C_{01}$ , which are replaced by cluster  $C_{10}$  at the first time interval.

As it can be seen in Figure 10.4 the number of clusters is not changed at the second and third time windows. However, one can easily observe that behavior profile  $C_{11}$  evolves its shape over these two time intervals. For example, it moves far from  $C_{21}$  and gets closer to  $C_{20}$  at the third time window (see Table 10.1). These observations are also supported by the heatmaps plotted in Figure 10.5. One can observe that the respective cells in the heatmap plotted in Figure 10.5 (right) have changed their color in comparison with the heatmap in Figure 10.5 (left).

We can trace the evolution of the clusters at each step of our algorithm by comparing the sets of cluster centroids of each pair of clusterings extracted from the corresponding consecutive time intervals. This is demonstrated in Figure 10.6 which plots the DTW alignment path between the clustering models generated on the historical and first new data sets, respectively. This comparison can be performed on any pair of clustering models generated on the studied data sets. It is also possible to track back the evolution of given final cluster down to the initial clustering model.

## 10.4 Conclusions and Future Work

In this paper, we have proposed a sequential partitioning algorithm that groups distinct snapshots of streaming data so that a clustering model is generated on each data snapshot. It enables to deal with both incremental and dynamic aspects of modeling evolving behavior problems. In addition, the proposed approach is able to trace back evolution through the detection of clusters' transitions. We have initially evaluated our algorithm on household electricity consumption data. The obtained results have shown that it is robust to modeling evolving data behavior by being enable to mine changes and adapt the model, respectively.

For future work, we aim to further study and evaluate the proposed clustering algorithm on evolving data phenomena in different application domains.

## References

- [1] C. C. Aggarwal. “On change diagnosis in evolving data streams”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.5 (2005), pp. 587–600.
- [2] A. Bouchachia. “Evolving clustering: An asset for evolving systems”. In: *IEEE SMC Newsletter* 36 (2011), pp. 1–6.
- [3] M. Oliveira and J. Gama. “A framework to monitor clusters evolution applied to economy and finance problems”. In: *Intelligent Data Analysis* 16 (1 2012), pp. 93–111.
- [4] M. Bottcher, F. Hoppner, and M. Spiliopoulou. “On exploiting the power of time in data mining”. In: *Proceedings of SIGKDD Explorations*. 2008, pp. 3–11.
- [5] A. Bouchachia. “Evolving clustering: an asset for evolving systems”. In: *IEEE SMC Newsletters* 36 (2011).
- [6] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving Systems* 3 (3 2012), pp. 135–151.
- [7] M. Ackerman and S. Dasgupta. “Incremental clustering: The case for extra clusters”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 307–315.

- 
- [8] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. “Incremental clustering and dynamic information retrieval”. In: vol. 33. 6. SIAM, 2004, pp. 1417–1440.
  - [9] M. Zopf and et al. “Sequential Clustering and Contextual Importance Measures for Incremental Update Summarization”. In: *Proc. of COLING'2016*. 2016, pp. 1071–1082.
  - [10] R. Fa and A. K. Nandi. “Smart: Novel self splitting-merging clustering algorithm”. In: *European Signal Processing Conference*. IEEE, 2012.
  - [11] M. Wang, V. Huang, and A. C. Bosneag. “A Novel Split-Merge-Evolve k Clustering Algorithm”. In: *IEEE 4th Int. Conf. on Big Data Comp. Service and Appl.* 2018.
  - [12] V. Boeva, M. Angelova, and E. Tsiporkova. “A Split-Merge Evolutionary Clustering Algorithm”. In: *Proceedings of ICAART 2019*. 2019, pp. 337–346.
  - [13] J. MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *5th Berkeley Symp. on mathematical statistics and probability*. Vol. 1. 14. 1967, pp. 281–297.
  - [14] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.
  - [15] C. Nordahl, V. Boeva, H. Grahn, and M. Netz. “Profiling of Household Residents’ Electricity Consumption Behavior using Clustering Analysis”. In: *Proceedings of International Conference on Computational Science*. Faro, Algarve, Portugal: Springer, June 2019.
  - [16] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
  - [17] J. Handl, J. Knowles, and D. B. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.
  - [18] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.



# EvolveCluster: An Evolutionary Clustering Algorithm for Streaming Data

---

*Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz*

*In: Evolving Systems 13(4), 2022, pp. 603-623. DOI: s12530-021-09408-y*

## Abstract

Data has become an integral part of our society in the past years, arriving faster and in larger quantities than before. Traditional clustering algorithms rely on the availability of entire datasets to model them correctly and efficiently. Such requirements are not possible in the data stream clustering scenario, where data arrives and needs to be analyzed continuously. This paper proposes a novel evolutionary clustering algorithm, entitled EvolveCluster, capable of modeling evolving data streams. We compare EvolveCluster against two other evolutionary clustering algorithms, PivotBiCluster and Split-Merge Evolutionary Clustering, by conducting experiments on three different datasets. Furthermore, we perform additional experiments on EvolveCluster to further evaluate its capabilities on clustering evolving data streams. Our results show that EvolveCluster manages to capture evolving data stream behaviors and adapts accordingly.

## 11.1 Introduction

In recent years, data has become an integral part of our daily lives. Due to advances in hardware infrastructures, there are endless possibilities available to collect any type of data at a rapid pace. Examples of streaming data sources include weather sensors, mobile applications, Instagram posts, electricity consumption, shopping records, etc. [1].

These data streams are endless information sources that arrive in a timely fashion. Incoming data tends to be unlabeled, as it requires too much effort to label it by hand. The immense amount of data proves to be hard to manage with traditional supervised machine learning algorithms and caused the emergence of unsupervised learning techniques. Unsupervised learning is a branch of machine learning where algorithms learn by themselves, identifying the underlying structure of a dataset. Depending on the application, the results from the unsupervised learning algorithms can be used directly for analysis or as an intermediary step to gain an understanding of the data. One of the branches of unsupervised learning is the task of clustering analysis.

Clustering is the process of grouping data instances into groups based on their similarity to each other [2]. Intuitively, instances within a cluster are more similar to each other than to other instances belonging to another cluster [3, 4]. The objective of clustering algorithms is to detect these underlying characteristics of the instances that make each cluster unique. Traditional clustering algorithms, such as  $k$ -means [5], require the entire dataset to be available. In data stream clustering, the data arrives incrementally in such a high quantity and pace that traditional clustering methods cannot cope with [2].

In the evolving data stream scenario, we have a continuous data stream that contains changes over time. These changes cause traditional offline models to become obsolete over time as the new data no longer conforms to how the model has been trained. Incremental clustering algorithms, such as the one introduced by [6], are one way to address the problem of evolving data streams. These algorithms process elements on a step-wise basis and injects them into the existing clustering solution, updating the clusters by merging or splitting if needed.

Many applications do, however, not necessarily need such rapid adaptations given by incremental clustering algorithms. Instead, by approaching the data stream segmentally, it is possible to view how the data is changing over time directly. For example, electricity providers can identify if and how the electricity consumption trend has altered in a single household, neighborhood, or an entire city, and determine if any remediation is required. Likewise, an online retailer can identify consumer shopping trends and how they change over time. When an overarching view of how the data aligns with previous structures and how it changes over time is desirable, there is

a lesser need for direct updates to the clustering models.

In this study, we propose a novel evolutionary clustering algorithm capable of modeling data streams containing evolving data, entitled EvolveCluster, a continuation of our previous work [7]. Instead of processing elements individually, we collect data over a defined period (segments) to trace how the data evolves. Two similar approaches have been identified to compare and evaluate the proposed algorithm, namely PivotBiCluster [8] and Split-Merge Evolutionary Clustering [9]. Both these algorithms address the evolving data stream scenario by dividing the data into segments. In contrast to EvolveCluster, PivotBiCluster and Split-Merge Clustering map previous clustered data segments to fit with the newly arrived data segment. Both these algorithms combine the current data segment with the previous one by identifying similarities between the clusters from the two segments. However, the main drawback with both PivotBiCluster and Split-Merge Clustering is that they both require each data segment to be clustered in advance.

Our main contributions are as follows:

- We introduce a new algorithm, entitled EvolveCluster, that is especially targeted at evolving data streams. The design of the algorithm makes it easy to understand how trends and patterns appear in the data segments (Section 11.4.2).
- We provide a thorough analysis of the computational complexity of the proposed algorithm (Section 11.4.3).
- We evaluate the performance of EvolveCluster, PivotBiCluster, and Split-Merge Clustering, and we identify and discuss their strengths and weaknesses (Section 11.6 and Section 11.7).

## 11.2 Background

In this section we provide the necessary background information. First a description of clustering analysis is provided, with a specific definition of  $k$ -medoids. We continue by introducing dissimilarity measures, and conclude this section with an explanation of evaluation and validation measures.

### 11.2.1 Clustering Algorithms

Clustering algorithms are designed to identify an underlying structure of data and use the detected relationships within the structure to group the data points into distinct groups. These algorithms usually decide upon themselves how to divide the data into subgroups, an unsupervised approach to increase knowledge about the data. There are numerous ways of approaching this task and we can group them into five major categories: density-based, grid-based, hierarchical, model-based, and partitioning algorithms [10]. This study focuses on partitioning algorithms due to the proposed evolutionary clustering algorithms characteristics (see Section 11.4).

Partitioning algorithms differ from the other algorithm types in their need to define the number of clusters in advance. The number of clusters, usually denoted as  $k$ , is a parameter given to the algorithms when they are initialized. But, identifying an appropriate  $k$  in advance is not easy. A common approach to identify a suitable  $k$  is having the algorithm execute multiple times with an increasing  $k$  value. More sophisticated methods exist, where the data is analyzed in advance with an initialization algorithm that estimates how many clusters are present in the dataset [11]. These initialization algorithms, however, do not promise to produce an optimal solution.

One of the most prolific examples of a partitioning algorithm is the  $k$ -means algorithm.  $k$ -means starts by assigning  $k$  initial cluster centroids, either randomly or by an initialization algorithm. All data points are distributed into each cluster based on their distance to the centroids. The solution is refined by first electing a new cluster centroid, based on the mean values of each data object in the cluster, and then redistributing the data points accordingly.  $k$ -means refines the solution until changes are no longer made or until a maximum limit of iterations has been reached.

$k$ -medoids, or Partitioning Around Medoids [12], is similar to  $k$ -means and generally seen as a sister algorithm.  $k$ -medoids, however, use actual data points as cluster centroids instead of creating synthetic centroids. This approach makes the algorithm more robust than  $k$ -means, being less susceptible to noise and outliers.

### 11.2.2 Concept Drift

One of the phenomena present in data streams, especially in evolving data streams, is how the data changes and evolves. This non-stationarity of data over time is referred to as concept drift [13]. Depending on the data and how it changes over time, different types of concept drifts may exist in the streams. [14] divided concept drift in six categories: sudden, incremental, gradual, recurring, blip, and noise. Sudden concept drifts are abrupt changes to the data, while incremental and gradual drifts happen more slowly. A recurring drift is a sudden, gradual, or incremental drift that happens periodically. Blip and noise are defined as outliers and random instances that should be filtered out, respectively.

Concept drift is a crucial aspect of learning from evolving data streams. As the data evolves, the algorithm needs to be capable of adapting and continuously learn about the underlying data structure to model it correctly. In addition to our comparative experiments (Sections 11.6.1-11.6.3), we perform an additional set of experiments to focus solely on EvolveCluster's ability to model data streams where concept drift is present (see Section 11.6.4).

### 11.2.3 Dissimilarity Measures

Calculating the distance, or dissimilarity, between two objects is a requirement to enable the use of distance-based clustering algorithms such as  $k$ -medoids [12]. These measures provide a numerical value that indicates how dissimilar or distant two data objects are. Numerous variants of measures exist, and their usage depends on the data itself. Two of the most common measures are the  $L_1$  and  $L_2$ , commonly referred to as Manhattan and Euclidean distance (ED) [15], respectively. These two measures are relatively simple and tend to be very effective when the dataset has a lower dimensionality.

Based on the application, a variety of dissimilarity measures exist. Concerns such as dimensionality, computational efforts, type of data, etc., factor in choosing the measure [16]. For instance, an electricity consumption dataset is a time series dataset. If the shape of the consumption, i.e., behavior, is desired, a measure such as DTW [17] is an eligible candidate measure. As an elastic measure, DTW could identify similar behaviors that occur at different times of day as closely related. If instead a strict measure was used, such as ED, those similar behaviors would likely not be identified as similar.

However, if there was a concern that the behaviors should be performed at the exact time and place each day to be identified as similar, ED would be a better choice of measure [18].

The datasets used in this study vary quite distinctively in their type and number of data points. None of them, however, is considered to be a high-dimensional dataset. Thus, we decided to use ED [15] on our datasets to focus on the algorithms and their properties. ED is defined as follows

$$\text{ED}(q, p) = \sqrt{\sum_{i=0}^n (q_i - p_i)^2}, \quad (11.1)$$

where  $q$  and  $p$  are two data vectors consisting of  $n$ -dimensions and  $p_i$  and  $q_i$  are individual points in  $p$  and  $q$ , respectively.

#### 11.2.4 Cluster Validation Measures

The data mining literature provides a wide range of different cluster validation measures, which are broadly divided into two major categories: external and internal [19]. External validation measures have the benefit of providing an independent assessment of clustering quality, evaluating the clustering result with respect to a pre-specified structure. Within the external evaluation, there are two distinct classes of measures: unary and binary [20]. Unary measures often take a clustering solution as input and compare it against the ground truth. The clustering solution can be evaluated with regard to both the purity and the completeness of the clusters.  $F_1$  is one example of such a validation measure [21]. In addition, to unary measures, a number of indices that assess the consensus between two partitioning solutions, based on the pairwise assignment of data points, are provided in the data mining litterature. Most of these indices are symmetric, making them well-suited for assessing the similarity of two clustering solutions, in which the Jaccard Index is a good example of [22].

Internal validation techniques, on the other hand, avoid the need for using such additional knowledge. They evaluate the clustering solutions based upon the same information that were used to create the clusters, enabling them to evaluate the quality of the produced clustering solutions in different ways. Internal measures can be divided into four categories based on how they evaluate clustering solutions: compactness, separation,

connectedness, and stability of the cluster partitions. A detailed overview of different types of validation measures is available in [23, 24].

Traditionally, many researchers working in data stream clustering apply well known validation measures to evaluate the clustering solutions produced by their data stream clustering algorithms [25], such as Sum of Squared Errors (SSE) and purity. Specific validation measures do, however, exist in the realm of data stream clustering. In 2011, Cluster Mapping Measure (CMM) is proposed as an effective measure for data stream clustering [26]. CMM is a combination score of missed objects, misplaced objects, and noise inclusion, and is based on the ground truth. More recently, several adaptations of well-known validation measures have been proposed, including Silhouette Index [27], Davies-Bouldin [27], and Xie-Beni [28]. All of the aforementioned measures are, however, designed for incremental clustering algorithms. The algorithm proposed in this paper divides the stream into fixed-sized segments and separates the segments to analyze and evaluate them individually. Due to the algorithm’s intended application and design, that a stream is divided into segments, it can be argued that it is not necessary to adopt the incremental validation measures. EvolveCluster does not process elements incrementally. Instead, each segment is statically clustered, which allows us to utilize traditional validation measures on each segment. Furthermore, as the algorithm only operates on entire segments and not individual instances, there is no directly applicable way to validate with these types of measures.

In the coming sub-sections, we describe and define the evaluation measures we apply in the study. We use two external ( $F_1$ -measure and Jaccard Index) and two internal (Silhouette Index and Average Intra-Cluster Distance) cluster validation measures to the clustering solutions generated by our experiments.

#### 11.2.4.1 $F_1$ -measure

$F_1$  is the harmonic mean of the precision and recall values of each cluster. Consider two clustering solutions,  $A = \{A_1, \dots, A_k\}$  and  $B = \{B_1, \dots, B_l\}$ , of the same dataset. We define  $A$  as the known partitioning of the dataset and  $B$  as the partitioning produced by the applied clustering algorithm. We

then define the  $F_1$  for a cluster  $B_j$  as:

$$F_1(B_j) = \frac{2|A_i \cap B_j|}{|A_i| + |B_j|}, \quad (11.2)$$

where  $A_i$  is the cluster containing the maximum number of objects from  $B_j$ .

To evaluate the overall  $F_1$  score for the clustering solution  $B$ , two common approaches are used, micro and macro average. Both versions are similar, but the macro average sees all classes as equal while the micro average corrects the score by each individual class's frequency. In this study, the datasets used (see Section 11.5.1.1) have a fairly even distribution of the corresponding classes. Therefore, the macro  $F_1$  is used and for the clustering solution  $B$  it is defined as:

$$F_1(B) = \frac{1}{l} \sum_{j=1}^l F_1(B_j), \quad (11.3)$$

where  $l$  is the number of clusters within  $B$ . The  $F_1$  score has a value between 0 and 1, with 1 indicating a perfect score.

#### 11.2.4.2 Jaccard Index

For evaluating the stability of a clustering solution, Jaccard Index (JI) is a suitable candidate. Given two clustering solutions produced from the same dataset,  $A$  and  $B$ , we define JI between  $A$  and  $B$  as follows:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (11.4)$$

where  $|A \cap B|$  is the number of data points with the same class in the same clusters in  $A$  and  $B$ , and  $|A \cup B|$  is the total number of data points in the same clusters in  $A$  and  $B$ . JI ranges from 0 to 1, where a higher value indicates a higher similarity between the clustering solutions.

#### 11.2.4.3 Silhouette Index

Silhouette Index (SI) is a cluster validity index that is used to determine the quality of any clustering solution  $C = \{C_1, \dots, C_k\}$ . It produces a score that is based on the compactness of each cluster and the separation between the clusters [29]. SI for a clustering solution  $C$  is defined as:

$$SI(C) = \frac{1}{m} \sum_{i=1}^m \frac{b_i - a_i}{\max(a_i, b_i)}, \quad (11.5)$$

where  $a_i$  is the average distance from object  $i$  to the other objects in its cluster and  $b_i$  is the minimum average distance from  $i$  to the objects of the other clusters. SI ranges from -1 to 1, where a value closer to 1 indicates a better clustering solution, and a value on the negative side of the range indicate that there are misplaced data points within the clustering solution.

#### 11.2.4.4 Average Intra-Cluster Distance

Similarly to SI, the Average Intra-Cluster Distance (IC-av) measures how compact the produced clusters are. In contrast to SI, it does not assume a spherical shape of the produced clusters [30]. Instead of calculating the radius around the clusters, IC-av produces a Minimum Spanning Tree (MST) of all data points based on the distance between the objects in the dataset. The edges containing the distance between the data points in the tree are then used to determine the compactness of the clusters in the clustering solution. For a particular clustering solution  $C = \{C_1, \dots, C_k\}$ , IC-av is defined as:

$$\text{IC-av}(C) = \sum_{r=1}^k \frac{1}{n_r} \sum_{i,j \in C_r} d_{ij}^2, \quad (11.6)$$

where  $n_r$  is the number of objects in cluster  $C_r$  ( $r = 1, 2, \dots, k$ ) and  $d_{ij}$  is maximum edge distance which represents the longest edge in the path joining objects  $i$  and  $j$  in the MST. IC-av produces a score between zero and the maximum value of the edges in the MST and should be minimized.

## 11.3 Related Work

In this section, we provide a review of studies related to our work. At the end of the section, we specifically review the two algorithms PivotBiCluster and Split-Merge Evolutionary Clustering.

### 11.3.1 Evolving Data Streams

The data stream clustering scenario differs from traditional clustering because the data is usually not available in its entirety. Additionally, the data in the stream arrives at such a rapid pace and in large quantities that it is impossible to keep the data in the main memory [1, 2]. Traditional algorithms, such as  $k$ -means [5] and DBSCAN [31], rely on the entire dataset being present. A naive approach to apply traditional algorithms on data streams would be to

re-cluster the entire solution at each increment. However, this approach is unfeasible both in regards to time constraints and the resources needed by the algorithms [4, 32].

In addition to the quantity and rapidness of data in data streams, evolving data streams have the additional dynamics of non-stationarity data over time, also known as concept drift [13]. Multiple approaches have been investigated to capture the dynamic aspects of evolving data streams, including [33–39]. More specifically, [6] proposes an incremental algorithm, where each increment causes the affected cluster to be split and merged separately, which was further developed in [40]. Similarly, [41] extends the  $k$ -means algorithm to a dynamic incremental clustering algorithm. A common idea for capturing evolving data streams’ dynamic nature is to use incremental algorithms and add functionality to modify clusters by splitting and merging [38, 41]. In general, these algorithms are divided into two components: Online and Offline [4]. The online component of the algorithms produces micro clusters that stay up to date with each increment of data objects that arrives, and the offline component runs periodically to finalize the clustering solution based on the produced microclusters.

### 11.3.2 Window Based Models

Generally, within data stream clustering, especially in contrast to traditional clustering, it can be more efficient to focus on the recent data instead of the entire stream. Several window models exist, but the following three are the most popular: damped window, sliding window, and landmark window [4]. The damped window models approach the data limitation by incorporating a weight factor when the data is processed. No object is removed from the window, but older the data objects have lower importance for the model. It is usually performed by a negative exponential function, such as  $f(t) = 2^{-\lambda t}$ . SNCStream [42] and its extension SNCStream+ [43] operate in a damped window scenario in a single pass manner. They are based on social network theory and use homophily to identify non-hyper spherical clusters. Similarly, pcStream [44] is also defined to operate in a damped window mode. pcStream dynamically detects and manages temporal contexts by fusing sensor data streams to infer the present concepts and detects new concepts as they emerge.

The sliding window models approach the data stream in a similar way as

damped windows but can be seen as stricter. Instead of having a decaying function, the sliding window is of a fixed size, and all objects within the sliding window have the same level of importance. When an object is added at one end of the window, another object is removed at the other end of the window, providing a window sliding over the stream.

DenStream [45] and its extension HDDStream [46], that handles high-dimensional data, follow an online-offline design, and are based on the DBSCAN clustering algorithm. The online procedures of the algorithms produce micro-clusters based on the density, which are later fed to the offline procedures, where the real clusters are created. WCDS [47] also follows the online-offline approach, creating micro-clusters in the online phase, but the offline phase was based on an agglomerative clustering algorithm to define its top-level clusters.

In contrast, landmark window models divide the data by assigning fixed landmarks where all data between two landmarks is a window. When a landmark is reached and a window ends, the succeeding window starts from that landmark point. A typical approach for landmark window-based clustering algorithms is to use the divided data stream to cluster them separately and use the produced centroids for that segment as representation, usually with partition-based algorithms.

The Stream framework was one of the earliest methods for stream clustering [48]. The data stream is divided into segments, and each segment is clustered by  $k$ -median. The produced cluster centers from the segments are added into buckets representing a prototype array, ending up with  $k_i$  medians, where  $i$  is the number of clustered segments. Whenever the number of stored medians surpass a parameter  $m$ ,  $k$ -medians is run upon the prototype array to produce a median of medians situation. Stream LSearch is an extension of the Stream method, where a more effective subroutine for the underlying  $k$ -median was introduced called LSearch [33]. In 2015, a stream adaptation called StreamKM++ [49] was proposed to the kmeans++ algorithms. StreamKM++ creates a coresset tree by sampling a subset of the segment and solves the optimization problem on that subset without touching the rest of the segment. These sets are then stored in buffers that are merged whenever a new segment is clustered. StreamXM is a continuation of StreamKM++ and operates similarly, with Xmeans as the underlying clustering algorithm [49].

DUCstream also divides the stream into segments that are manageable for the system memory, but its underlying structure is instead a density-based algorithm [50]. DUCstream partitions the data space in units and map the incoming objects in the units; the more mapped objects to a unit, the denser it is. These dense units are then used to perform clustering.

None of the methods mentioned above are explicitly tailored for our specified problem. They aim to model the entire stream with a single clustering solution as best as possible. We instead intend to divide the stream into segments, cluster them separately, and use the clustered segments to see how the stream evolves. With clustering solutions produced of each segment, it is easier to analyze the data between segments and trace how clusters have remained, changed, disappeared, or appeared. We have identified two approaches that similarly address the data stream clustering problem, namely PivotBiCluster [8] and Split-Merge Evolutionary Clustering algorithms [9].

### 11.3.3 PivotBiCluster

The first algorithm we compare with is PivotBiCluster [8], an algorithm related to Bipartite Correlation Clustering (BCC) [51]. BCC builds upon the notion of taking two clustering solutions and combine them into a larger solution. Either by directly combining two clusters from different solutions or dividing a cluster from one solution into several clusters in the other solution. The combination is decided upon the correlation between the clusters of the different clustering solutions.

Referring to our problem statement, located in Section 11.4.1, PivotBi-Cluster assumes two data segments have been clustered beforehand, e.g.,  $D_0$  and  $D_1$ , thus producing  $C_0$  and  $C_1$ . These two clustering solutions ( $C_0$  and  $C_1$ ) are then given to PivotBiCluster, which tries to combine them together, creating  $C'_1$ , by merging clusters from each solution based on how similar they are to each other. The correlation clustering can be applied over and over; thus, in the formalized problem statement, the PivotBiCluster continues to create a large clustering solution by using  $C'_1$  in combination with  $C_2$  to produce  $C'_2$ .

One of the drawbacks of the PivotBiCluster algorithm is a lack of the ability to split a cluster into several others in the other clustering solutions. This drawback was the primary motivation of the Split-Merge Evolutionary Clustering algorithm.

### 11.3.4 Split-Merge Evolutionary Clustering

The Split-Merge Evolutionary Clustering (Split-Merge Clustering) algorithm builds upon the idea present in BCC clustering algorithms, with the addition of splitting a cluster into multiple clusters in the other clustering solution [9]. This means that if a larger cluster exists within one of the clustering solutions, it can be split up into multiple clusters in the algorithm's output. Similar to the approach of PivotBiCluster, the Split-Merge Clustering algorithm also assumes the incoming data from the data stream  $D$  is clustered in advance. Additionally, just as the PivotBiCluster, the clustering can either occur continuously, thus create a final large clustering solution that contains all the data elements from the dataset. Another option can be to take intermediary steps to filter out the data from the previous segment(s) to create a more reflecting model on the present type of data in the data stream.

One of the significant benefits of Split-Merge Clustering, compared to PivotBiCluster, is the splitting of clusters. The authors claim that with that addition, the algorithm is less sensitive to under- and over-clustering of each data segment, as the clusters are now more easily modified over time.

In contrast to our proposed algorithm, which we present in the following section, the Split-Merge Clustering algorithm links the old and new clustering solutions together. Depending on what type of data is being analyzed, this can be counter-productive if the clustering aims not to focus on current trends in the data solely.

## 11.4 An Evolutionary Clustering Algorithm

### 11.4.1 Problem Statement

Let us formalize the evolving data scenario we aim to address. Assume that  $D$  is a continuous stream of data, and a vector of features represents each data point.  $D_0$  is the initial data segment which has been partitioned into  $k$  clusters,  $C_0 = \{C_{00}, \dots, C_{0k}\}$ . Additionally,  $D_1, \dots, D_t$ , where  $t \rightarrow \infty$ , are continuous segments of data in the stream to be partitioned. Our objective is to produce a clustering solution, or clustering solutions, modeling how the data evolves.

### 11.4.2 EvolveCluster: An Evolutionary Clustering Algorithm

In this section, we formally describe the proposed sequential partitioning algorithm, entitled EvolveCluster. The main idea of EvolveCluster is to allow a continuous data behavior to be easily modeled, by incorporating gained knowledge from the previous data segments, in the form of the cluster centroids, to influence the clustering of the new data segment. Using previous centroids, we can trace how the clusters evolve as the clusters are related over the data segments. Likewise, with each segment being clustered individually, it is easy to identify reoccurring trends between segments and changes in the data. The algorithm idea is schematically illustrated in Figure 11.1.

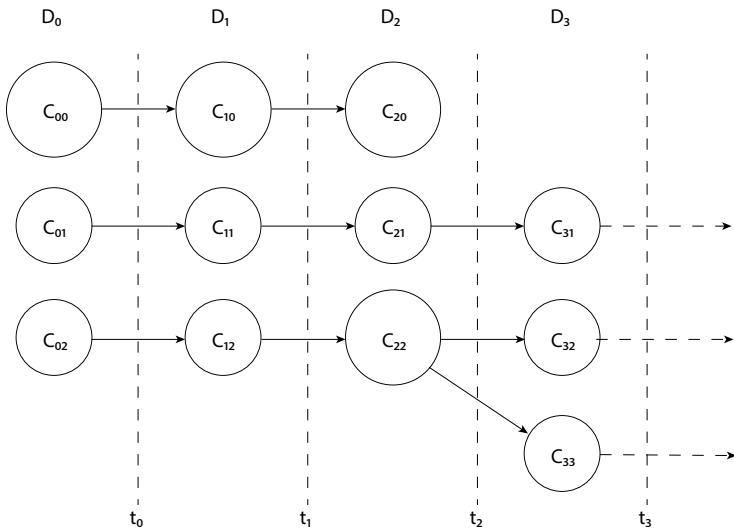


Figure 11.1: A schematic illustration of the proposed EvolveCluster.  $D_i$  represents data segments at time  $t_i$ ,  $C_{ij}$  represents clusters in clustering solutions  $C_i$  of data segments  $D_i$ , and  $t$  represents individual time segments.

Similar to both PivotBiCluster and Split-Merge Clustering, EvolveCluster divides the data stream into individual data segments. Likewise, the initialization of EvolveCluster requires the first data segment to be clustered in advance. The remaining data segments are, however, clustered within EvolveCluster. Each segment is sequentially clustered, using a partitioning algorithm, with the aid of the cluster centroids (seeds) from the previous segment. EvolveCluster assumes the new data segment contains at least

some of the structure from the previous segment by incorporating the clustering structure from the previous segment. The following operations are conducted on each new data segment:

- The data points of the segment is initially clustered by seeding with the cluster centroids of the previous segment;
- The old centroids are removed and any empty clusters are deleted;
- New centroids for the clusters are elected, and the clustering solution is refined.

The refined clustering solution undergoes a “trial-and-error” approach to detect if any clusters should be split into two by applying a 2-means clustering algorithm on a cluster basis. The 2-means clustering algorithm is initialized with the two data points in the cluster that exhibit the furthest distance to each other. If the clustering solution containing the split clusters is deemed the better clustering solution, by a validation measure, it is kept. Otherwise, it is discarded. The algorithmic steps conducted at each data segment are defined in Algorithm 1.

### 11.4.3 Computational Complexity

In this section, we examine the computational costs of the clustering and splitting operations of the proposed algorithm. Depending on what underlying clustering algorithm is used, the computational complexity will differ. The approach proposed in this study uses  $k$ -medoids, a distance based partitioning algorithm.  $k$ -medoids requires a distance matrix of size  $n \times n$  to be computed, where  $n$  is the number of elements. The distance matrix occupies the majority of both the computations and memory consumption of the algorithm, being a complexity of  $O(n^2d)$  and  $O(n^2)$ , respectively, where  $d$  is the feature space dimension.

In this study, we propose the use of  $k$ -medoids whose complexity has been thoroughly studied [52]. We can divide  $k$ -medoids into two parts: i) Initialization and ii) Refinement. The initialization according to the original implementation, which opts to identify a beneficial starting point, generates a complexity of  $O(n^2k)$  where  $k$  is the number of clusters. When initial medoids are provided, or randomly chosen, the complexity instead becomes  $O(nk)$ . However, the refinement process remains the same as originally

**Algorithm 1:** EvolveCluster

---

```

Input : Data segment  $D_t$ , Centroids  $\{c_1, \dots, c_k\} \in C_{t-1}$ 
Output: Clustering solution  $C_t$ 

1  $C_t \leftarrow \text{InitialPartition}(D_t, c_1, \dots, c_k)$ 
2  $C_t \leftarrow \text{RefineSolution}(C_t)$ 
3  $S \leftarrow \text{SI}(C_t)$ 
4 while SplitPerformed = True do
5   SplitPerformed  $\leftarrow$  False
6   for  $C_{ti} \in C_t$  do
7      $C'_t \leftarrow \text{Split}(C_t, C_{ti})$ 
8     if  $S + \tau < \text{SI}(C'_t)$  then
9        $S \leftarrow \text{SI}(C'_t)$ 
10       $C_t \leftarrow C'_t$ 
11      SplitPerformed  $\leftarrow$  True
12      break
13    end
14  end
15 end

```

---

defined, generating a complexity of  $O((n - k)^2 ki)$ , where  $i$  is the number of iterations performed in the refinement. This we can simplify to  $O(n^2ki)$  as  $k \ll n$ .

Here, we present the computational complexity of a single iteration of EvolveCluster. Suppose  $n$  is the number of data instances in the entire dataset and  $n'$  is the number of instances in each data segment, where  $n' \ll n$ . The initial clustering occurs in two steps, InitialPartition and RefineSolution, as defined in Algorithm 1 (steps one and two, respectively). InitialPartition assigns each data object in the current segment to the closest centroid, removes the initial centroids, and deletes any empty clusters, with a computational cost of  $O(n'k + k + k) \rightarrow O(n'k)$ . RefineSolution is a direct implementation of the original  $k$ -medoids algorithm, giving a complexity of  $O(n'^2ki)$ . The initial clustering of EvolveCluster then becomes  $O(n'k + n'^2ki)$ , which can be simplified to  $O(n'^2ki)$ .

The split criterion of EvolveCluster is calculated once outside the loop and once for every time a split is performed inside the loop. In the proposed approach, we use the SI as our measure for the split criterion, which has a

computational complexity of  $O(2n' + n'^2)$ . EvolveCluster splits a cluster by first identifying the two elements that are the furthest apart,  $O(n'^2)$ .  $k$ -medoids is then used with  $k = 2$  with the two identified elements as initial centroids, i.e.  $O(n'k + n'^2ki)$ . A single iteration of the splitting loop then becomes  $O(n'^2 + n'k + n'^2ki + 2n' + n'^2)$ , which we can simplify to  $O(n'^2ki)$ .

As each cluster in the produced clustering solution  $C_t$  has to be split at least once, the lower bound of iterations for the splitting part of EvolveCluster becomes  $k$  times. This gives the lower bound for splitting to be  $O(k(n'^2ki)) \rightarrow O(n'^2k^2i)$ . The upper bound, on the other hand, is dramatically higher. In the worst case, a split is performed in every iteration which causes the final clustering solution to consist solely by singleton clusters, i.e.,  $n'$  iterations. The upper bound then becomes  $O(n'(n'^2ki)) \rightarrow O(n'^3ki)$ . Finally, the total complexity for each increment, with the inclusion of the distance matrix calculation, of the EvolveCluster algorithm is  $O(n'^2ki + n'^2k^2i + n'^2d) \rightarrow O(n'^2(k^2i + ki + d)) \rightarrow O(n'^2(k^2i + d))$ . If we include the upper bound calculation, the complexity of EvolveCluster becomes  $O(n'^2ki + n'^3ki + n'^2d) \rightarrow O(n'^3ki)$ .

Similarly to EvolveCluster, the Evolutionary Split-Merge Clustering bases its complexity on the underlying clustering algorithm [9]. Evolutionary Split-Merge Clustering adds the additional computational overhead  $O((k' + k')n')$ . In combination with  $k$ -medoids, its complexity becomes  $O((k' + k')n' + n'^2ki) \rightarrow O(n'^2ki)$ , which is in line with the produced lower bound complexity of EvolveCluster. The authors of PivotBiCluster, on the contrary, have not proposed their complexity calculations in the clustering scenario [8]. Thus, we have no direct comparisons to perform.

## 11.5 Data and Experimental Designs

We perform two sets of experiments to investigate the effectiveness of EvolveCluster. The first experiment compares EvolveCluster and two similar clustering algorithms on three different datasets to analyze their differences and performances. In our second experiment, we analyze how EvolveCluster handles different concept drift scenarios by generating a synthetic data stream.

### 11.5.1 Experiment 1: Comparative Analysis

#### 11.5.1.1 Data

We evaluate and compare the performance of the proposed EvolveCluster algorithm to two other clustering algorithms (PivotBiCluster and Split-Merge Clustering) on three different datasets, explained in Table 11.1. The first is the S1 dataset, a 2-dimensional synthetic dataset created by the authors of [53]. This dataset is chosen to investigate the algorithms ability to identify new clusters as they arrive in the data stream, and how they manage with regard to clustering a constant type of behavior over time.

The second dataset is a subset of the Covertype dataset, available at the UCI repository [54]. The motivation behind the use of this dataset is mainly to have a direct comparison to both the PivotBiCluster and Split-Merge Clustering algorithms, as the authors of the latter algorithm have performed experiments upon it in their paper [9]. However, it is also chosen due to its larger number of data points in combination with a higher dimensionality of its features compared to the S1 dataset.

Finally, the third dataset is a real world electricity consumption dataset, the Domestic Electrical Load Metering, Hourly Data (DELMH) [55]. DELMH contains consumption from a large number of households and metering stations in South Africa covering the period from 1994 to 2014, with measurements taken up to every 5 minutes. It is worth noting that the single household with the most prolonged consumption period amounts to roughly two years worth of consumption. This type of dataset is the main area of target for our proposed algorithm.

All information about the used datasets in their original form is presented in Table 11.1.

Table 11.1: *Information regarding number of features and instances of each dataset in their original form. The number of instances in the DELMH dataset are individual measurements from 71 up to 2940 concurrent households over 21 years, varying between 1 measurement up to 12076 per household.*

Dataset	No. features	No. instances
S1	2	5000
Covertype	54	581012
DELMH	1	3341726

### 11.5.1.2 Data Pre-Processing

**S1 Dataset:** The S1 dataset is divided into 5 equal parts, each part consisting of 1'000 elements. We do, however, create two distinct experimental datasets from the S1 dataset. The first dataset keeps the original format where each cluster appears one by one in order, but the other dataset is modified such that all data segments contain the same ratio of all clusters (see below). The two features are normalized in the  $[0 - 1]$  range by a Min-Max feature scaling. Each feature value is subtracted by the minimum value of that feature ( $X_{min}$ ), and then divided by the difference between the minimum and maximum value ( $X_{max}$ ) of the feature, i.e.,

$$x' = \frac{x - X_{min}}{X_{max} - X_{min}}.$$

With the S1 dataset, we want to allow the algorithms to showcase how they handle two aspects of evolutionary clustering. The first is to discover new clusters as they appear in the data. By keeping the S1 dataset in its original state, each cluster appears in the data stream one after another. Between each segment in the original dataset, 2 to 3 clusters disappear, and 2 to 3 new clusters appear.

The second aspect is to model a continuous set of behaviors in the data stream over time. We simulate this aspect by dividing the data points in each cluster evenly between each segment. Thus, 20% of each clusters' data points are located in  $D_0$ .  $D_1$  consists of the next 20% appear and so on. From here on and forward, we denote this dataset as the *continuous S1* dataset.

**Covertype Dataset:** To mimic the experiments of the authors of the Split-Merge Evolutionary Clustering algorithm, we perform the same steps of pre-processing and segmentation of the Covertype Dataset [9]. A subset of 50'000 elements is randomly chosen out of the 581'012 and 14 out of the 54 features is chosen, excluding all binary features regarding the soil type. Each feature is standardized using the z-score, where each feature is subtracted by their mean value ( $\bar{x}$ ) and divided by the standard deviation ( $\sigma$ ), i.e.

$$z = \frac{x - \bar{x}}{\sigma}.$$

The 50'000 data points are divided into 2 segments in a 70-30 split, creating  $D_0$  with 35'000 elements and  $D_1$  with 15'000 elements.

Table 11.2: Information regarding number of features and data points in each data segment of all datasets after pre-processing.

Dataset	No. features	No. instances				
		D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
S1 original	2	1000	1000	1000	1000	1000
S1 continuous	2	995	1000	1000	1000	1005
Covertype	14	35000	15000	-	-	-
DELMH	24	198	74	75	74	75

**DELMH Dataset:** For the DELMH dataset, we first divide all available measurements into their corresponding households. All measurements are combined into daily profiles, such that each daily profile contains measurements from 00:00 to 23:59. Every daily profile consists of 24 data points, where each data point in the profile represents the aggregated consumption of each hour. If any profile contains a measure that is indicated to be invalid, the entire daily profile is dismissed for further use. All profiles undergo the same z-score standardization as mentioned above for the Covertype dataset. However, instead of applying it on a feature level, we apply it to each individual profile. Each standardized profile represents the shape of the electricity consumption and disregards the actual amplitude of the electricity consumption.

We identify the 10 households with the largest number of daily profiles, and choose one of them to represent the use case for our algorithm. The chosen household consists of 496 daily profiles after the pre-processing stage, starting from 1997-12-31 and ending on 1999-05-06. The final 496 profiles are divided into 5 segments, where the first segment ( $D_0$ ) contains 198 elements, corresponding to 40% of the number of profiles. The remaining 4 segments contain 74-75 profiles each, representing 15% of the total number of profiles.

A summary of all dataset information after the pre-processing and modification is located in Table 11.2.

### 11.5.2 Experiment 2: Concept Drift Analysis

In the second experiment, we specifically investigate how EvolveCluster performs in an evolving data stream scenario. This experiment is conducted on generated synthetic data to make sure a ground truth is available, the data contains concept drift, and when the concept drifts occur. We created a Radial Basis Function Generator (RBFGenerator) based on the implementations available at MOA [56] and scikit-multiflow [57]. Both MOA

and scikit-multiflow implementations provide evolving data streams that contain a constant drift of clusters, where each cluster centroid moves as time progress. However, scikit-multiflow's implementation does not contain any creation or deletion of clusters. Conversely, MOA includes options of specific events, such as cluster creation and deletion, but cannot export its stream if more than one additional cluster is created in the stream. Thus, we have created an RBFGenerator that produces evolving data streams with no limitation on the functionalities mentioned.

The produced data streams consist of 10'000 data points with 2 features. Each stream is initialized with the same random seed but has different seeds for generating data points and cluster events. When 2'500 data points have been produced, an event occurs in the stream. Additional events then occur after each 2'000th data point, i.e. at 4'500, 6'500, and 8'500. An event is randomly chosen out of two options, creation or deletion. The streams begin with 5 clusters and are allowed to vary between 2 and 8. The cluster centers are limited to the [0-1] domain in both features. The cluster centers' speed is randomly chosen but limited to 0.0001 per instance created in the stream. Similarly, the radius of each cluster is randomly chosen but limited to be  $0.02 \pm 0.005$ .

The data streams are divided into five segments each, creating data segments  $D_0, D_1, D_2, D_3, D_4$ . Each segment from  $D_1$  and onwards contains an event. To initialize EvolveCluster on the produced data stream, we used the cluster labels given by the RBFGenerator on the  $D_0$  segments and then calculated the centermost point (i.e., medoid) in each cluster to use as cluster centers.

### 11.5.3 Evaluation and Validation

In this study, we combine both internal and external cluster validation measures to assess the results from both experiments. In the first experiment, both the Covertype and the S1 datasets have ground truth labels, allowing us to use external validation measures. For these two datasets, we have used both the  $F_1$  and JI measures to evaluate how the three studied clustering algorithms perform regarding the known structure of the datasets. Additionally, we apply the SI to assess the compactness and separation of the produced clusters. The same applies to the data stream we generate for the second experiment; thus, we apply the same validation measures as

for Covertype and S1. The DELMH dataset in the first experiment, on the other hand, has no ground truth available, causing us to focus solely on the internal cluster evaluation measures. We instead apply the SI and IC-av measures to evaluate how good the produced clustering solutions are.

To show how the three algorithms in the first experiment perform over time, we calculate the evaluation metrics for each individual data segment of the datasets in three ways:

1. Only the data arriving in the current segment is used for calculating the scores, disregarding how the old data segment has been altered and merged.
2. Each segment is calculated in combination with the previous data segment.
3. For some of the experiments, we also evaluate all the data segments up until that point in time, e.g., when we reach  $D_3$ , we include  $D_0, D_1$  and  $D_2$  in the evaluation.

#### 11.5.4 Implementation and Availability

PivotBiCluster and Split-Merge clustering operate by taking existing clustering solutions and combine them into a larger clustering solution. Both algorithms require each data segment to be clustered beforehand and combine the produced clustering solutions to create a combined version. EvolveCluster, on the other hand, focuses solely on clustering the current data segment and only incorporates the cluster centroids of the previous data segment to produce the next, disregarding the old clustering solution after its initial clustering. To compare the results of these three algorithms, we have implemented two additional variations of the PivotBiCluster and Split-Merge Clustering algorithms. The first variation is implemented so that after each segment is clustered, data belonging to the previous data segment is removed, similar to the EvolveCluster algorithm. The second variation retains the previous data segment as the clustering progresses but is removed before the next data segment is clustered.

For the first experiment, all three clustering algorithms are initialized using the same clustering solution  $C_0$ . Providing all algorithms with the same starting point gives us insight into how they produce clustering solutions for

each data segment. For PivotBiCluster and Split-Merge Clustering, we have to cluster  $D_1, D_2, \dots, D_n$  before using them in the algorithms. In the S1 and Covertype experiments, we use the ground truth labels in conjunction with the NearestCentroid classifier [58] to find the centroids to be able to cluster  $C'_2$  and onwards. For DELMH, we employ  $k$ -medoids to do the initial clustering of  $C_1, C_2, \dots, C_n$ . This initial clustering is run for 1'000 iterations for each segment and the number of clusters between 2 and 10. Each clustering solution is evaluated via SI and IC-av. Empirically we choose upon the produced clustering solutions as the input for the PivotBiCluster and Split-Merge Clustering algorithms. Finally, all the experiments use the Euclidean Distance as the dissimilarity measure.

All experiments and algorithms are implemented in Python 3.6.10 and are available for download here<sup>1</sup>.

## 11.6 Results and Analysis

In this section the results from all experiments are presented and discussed following the order in which the datasets have been explained in Section 11.5.1.1.

### 11.6.1 Original S1 Dataset

#### 11.6.1.1 Original S1

The results from the experiment on the original S1 dataset are presented in Figure 11.2 and Table 11.3. In Figure 11.2, we observe that the EvolveCluster and Split-Merge Clustering algorithms are more proficient than PivotBiCluster in identifying new clusters when they arrive in the data stream. This is further strengthened in Table 11.3, where overall scores suggest that PivotBiCluster performs slightly worse compared to the other two algorithms. However, as it can be seen in Figure 11.2 a) (under the  $D_2$  header) we observe that EvolveCluster shows a difficulty in merging clusters together when they are initiated closely together. This result is logical, since EvolveCluster has no specific merge criterion or dedicated process for merging more than if the initial clustering of each segments produce empty clusters they are removed. It is also interesting to notice that Split-Merge Clustering fully follows the true clustering of the data points, up to the point that even data points that are overlapping into another cluster is correctly classified.

---

<sup>1</sup> <https://github.com/christiannordahl/EvolveCluster>

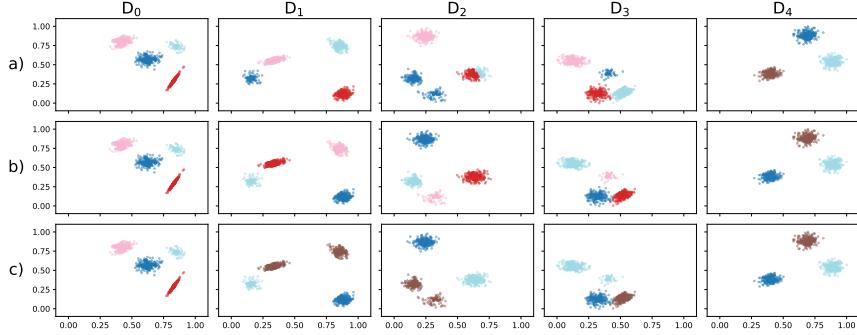


Figure 11.2: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  of the original S1 dataset. The results generated by the three studied algorithms are depicted as follows: a) EvolveCluster, b) Split-Merge Clustering, and c) PivotBiCluster. Each color of the data points in the figures represents a single cluster within that segment.

Table 11.3: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the original S1 dataset, where previous data segments are discarded before evaluating, for all three clustering algorithms.

		$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
EvolveCluster	$F_1$	1	1	0.770	0.997	1
	$J_1$	1	1	0.681	0.993	1
	$SI$	0.826	0.879	0.649	0.750	0.867
Split-Merge	$F_1$	1	1	1	1	1
	$J_1$	1	1	1	1	1
	$SI$	0.826	0.879	0.848	0.747	0.867
PivotBiCluster	$F_1$	1	0.908	0.947	0.978	1
	$J_1$	1	0.856	0.909	0.958	1
	$SI$	0.826	0.538	0.824	0.700	0.867

Table 11.4: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the original S1 dataset, where the previous data segment is kept during evaluation, for all three clustering algorithms.

		$D_0 - D_1$	$D_1 - D_2$	$D_2 - D_3$	$D_3 - D_4$
EvolveCluster	$F_1$	1	0.875	0.854	0.998
	$J_1$	1	0.827	0.797	0.995
	$SI$	0.783	0.716	0.567	0.793
Split-Merge	$F_1$	0.824	0.797	0.816	0.778
	$J_1$	0.735	0.687	0.750	0.705
	$SI$	0.313	0.365	0.129	0.103
PivotBiCluster	$F_1$	0.758	0.764	0.750	0.778
	$J_1$	0.672	0.679	0.664	0.705
	$SI$	0.414	0.176	0.222	0.103

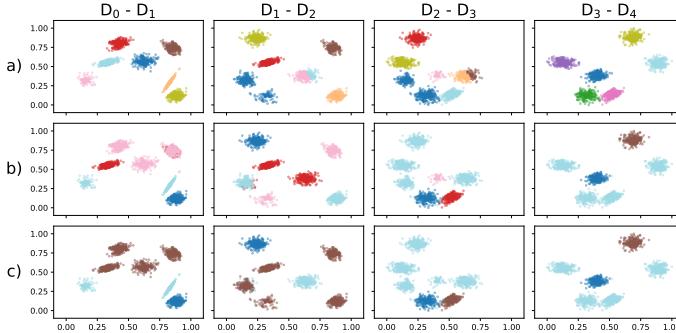


Figure 11.3: The clustering solutions obtained on each data segment  $D_1, D_2, D_3, D_4$  of the original  $S1$  dataset, where the previous segment is included. The results generated by the three studied algorithms are depicted as follows: a) *EvolveCluster*, b) *Split-Merge Clustering*, and c) *PivotBiCluster*. Each color of the data points in the figures represents a single cluster within that segment.

To further investigate how the three algorithms operates, we include the previous data segments for each clustering solution as explained in Section 11.5.3. These results are presented in Figure 11.3 and Table 11.4. It is clear that most of the structure from the previous data segment is lost when combining clustering solutions, both by the *PivotBiCluster* and, especially, the *Split-Merge Clustering* algorithm. *Split-Merge Clustering* has a perfect score in both  $F_1$  and  $JI$  when each segment is validated separately, which is to be expected. Each of the segments are clustered in advance and are following the ground truth labels from the dataset. Comparing the scores from the evaluation measures in Table 11.4, we also observe a significant decrease of all scores for all three algorithms. The drop for *EvolveCluster*, however, is minor in comparison.

Furthermore, in Table 11.5 we show the results for *PivotBiCluster* and *Split-Merge Clustering* when all previous segments are retained in the clustering solutions. It is clear that as the algorithms progress through the data segments, more clusters from the previous segments get merged into one large cluster for both *PivotBiCluster* and *Split-Merge Clustering*.

### 11.6.1.2 Continuous S1 Dataset

In this subsection, we present the results obtained from the continuous version of the  $S1$  dataset, presented in Figure 11.4 and Table 11.6. We observe that the performance of *PivotBiCluster* in this scenario dramatically

Table 11.5: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the original  $S1$  dataset, where the all previous data segments are kept during evaluation, for the PivotBiCluster and Split-Merge Clustering algorithm.

		$D_0$	$D_0 - D_1$	$D_0 - D_2$	$D_0 - D_3$	$D_0 - D_4$
Split-Merge	$F_1$	1	0.824	0.660	0.760	0.691
	$J_1$	1	0.735	0.534	0.712	0.646
	$SI$	0.826	0.313	0.121	-0.102	-0.166
PivotBiCluster	$F_1$	1	0.758	0.613	0.760	0.691
	$J_1$	1	0.672	0.564	0.712	0.646
	$SI$	0.826	0.414	0.267	-0.102	-0.166

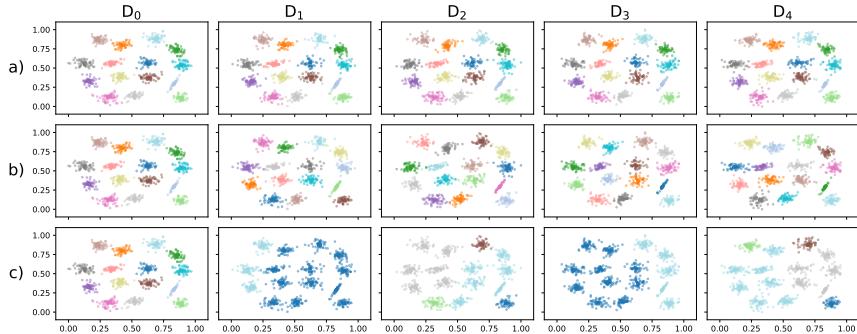


Figure 11.4: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  of the continuous  $S1$  dataset. The results generated by the three studied algorithms are depicted as follows: a) EvolveCluster, b) Split-Merge Clustering, and c) PivotBiCluster. Each color of the data points in the figures represents a single cluster within that segment.

Table 11.6: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the continuous  $S1$  dataset, where previous data segments are discarded before evaluating, for all three clustering algorithms.

		$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
EvolveCluster	$F_1$	0.997	0.990	0.993	0.996	0.992
	$J_1$	0.994	0.980	0.986	0.992	0.984
	$SI$	0.722	0.715	0.698	0.716	0.708
Split-Merge	$F_1$	1	1	1	1	1
	$J_1$	1	1	1	1	1
	$SI$	0.719	0.710	0.693	0.713	0.705
PivotBiCluster	$F_1$	1	0.334	0.720	0.269	0.722
	$J_1$	1	0.213	0.670	0.158	0.672
	$SI$	0.719	0.271	0.208	0.354	0.270

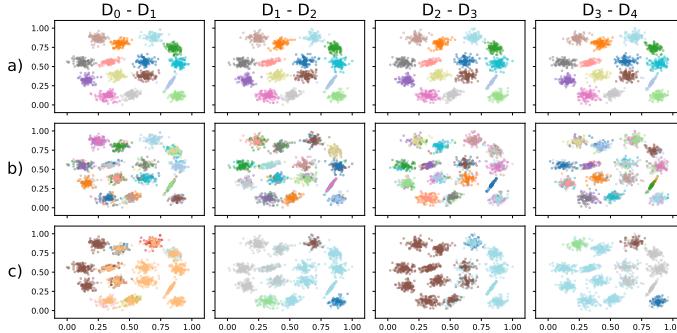


Figure 11.5: The clustering solutions obtained on each data segment  $D_1, D_2, D_3, D_4$  of the continuous S1 dataset, where the previous segment is included. The results generated by the three studied algorithms are depicted as follows: a) EvolveCluster, b) Split-Merge Clustering, and c) PivotBiCluster. Each color of the data points in the figures represents a single cluster within that segment.

decreases. In the first iteration, when  $D_1$  is clustered, there is an instant decrease in the number of clusters (see Figure 11.4). The clustering solution should contain 15 clusters, as is the case for both EvolveCluster and Split-Merge Clustering. PivotBiCluster instead opts to reduce the number of clusters to 2, showing a clear case of under-clustering. In contrast, both EvolveCluster and Split-Merge Clustering show that they can cluster a continuous set of behaviors over time.

Similarly to the original S1 dataset, when the data segments are merged, it is apparent that Split-Merge Clustering fully adapts the previous data segment's clustering solutions to the new segments. Figure 11.5 b) shows that many of the data points are incorrectly clustered for Split-Merge Clustering. PivotBiCluster has a similar experience when the segments are merged, as on the separated segments, a clear result of under-clustering. EvolveCluster, on the other hand, manages to retain a higher performance with validation measure scores on par with the non-merged segments. These results are further emphasized by looking at Table 11.7. Only EvolveCluster manages to produce similar results as when the segments are not merged. Both PivotBiCluster and Split-Merge Clustering show a drastic decrease in all three measures, with SI even producing negative numbers. Finally, when we include all the previous data segments in both the clustering and evaluation, as shown in Table 11.8, the produced clustering solutions of PivotBiCluster and Split-Merge Clustering are both continuing the declining trend.

Table 11.7: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the continuous  $S1$  dataset, where the previous data segment is kept during evaluation, for all three clustering algorithms.

		$D_0 - D_1$	$D_1 - D_2$	$D_2 - D_3$	$D_3 - D_4$
EvolveCluster	$F_1$	0.993	0.991	0.995	0.994
	$J_1$	0.987	0.983	0.989	0.988
	$SI$	0.718	0.707	0.707	0.711
Split-Merge	$F_1$	0.511	0.518	0.514	0.52057
	$J_1$	0.346	0.354	0.351	0.358
	$SI$	-0.113	-0.141	-0.121	-0.153
PivotBiCluster	$F_1$	0.564	0.517	0.396	0.514
	$J_1$	0.410	0.370	0.267	0.368
	$SI$	-0.109	-0.065	0.177	0.025

Table 11.8: Results from the validation measures  $F_1$ ,  $J_1$ , and  $SI$  on the continuous  $S1$  dataset, where the all previous data segments are kept during evaluation, for the PivotBiCluster and Split-Merge Clustering algorithm.

		$D_0$	$D_0 - D_1$	$D_0 - D_2$	$D_0 - D_3$	$D_0 - D_4$
Split-Merge	$F_1$	1	0.511	0.398	0.340	0.343
	$J_1$	1	0.346	0.251	0.207	0.210
	$SI$	0.719	-0.113	-0.228	-0.206	-0.146
PivotBiCluster	$F_1$	1	0.564	0.346	0.250	0.303
	$J_1$	1	0.410	0.214	0.143	0.180
	$SI$	0.719	-0.109	-0.067	0.227	-0.166

Figure 11.10 and Figure 11.11, available in the Appendix, show the results produced on both the original and the continuous  $S1$  datasets when all segments are combined.

### 11.6.2 Covertype Dataset

The results produced by the three clustering algorithms on the Covertype dataset are shown in Table 11.9. As can be seen, PivotBiCluster outperforms both the Split-Merge Clustering and EvolveCluster algorithms in the case of the  $D_0 - D_1$  setup in the table. These results are in line with the results presented in [9]. It is interesting to notice the difference between EvolveCluster and Split-Merge Clustering on what scores their clustering solutions obtain in the  $D_1$  column compared to the  $D_0 - D_1$  column. Evidently, EvolveCluster does not manage to cluster the  $D_1$  dataset as proficiently as the Split-Merge Clustering algorithm and the final score when the data segments are merged is helped by the clustering from  $D_0$ . Additionally, as the results generated on the  $S1$  datasets show (see Figures 11.3 and 11.5),

Table 11.9: Results from the validation measures  $F_1$ ,  $J_1$  and  $SI$  on the Covertype dataset for all three algorithms, on both the individual ( $D_0$  and  $D_1$ ) and combined ( $D_0 - D_1$ ) data segments.

		$D_0$	$D_1$	$D_0 - D_1$
EvolveCluster	$F_1$	1	0.422	0.539
	$J_1$	1	0.275	0.436
	$SI$	0.063	-0.007	-0.040
Split-Merge	$F_1$	1	1	0.754
	$J_1$	1	1	0.656
	$SI$	0.063	0.062	0.034
PivotBiCluster	$F_1$	1	0.905	0.903
	$J_1$	1	0.849	0.848
	$SI$	0.063	0.192	0.194

when the two data segments are combined, it is clear that the clustering solution on  $D_0$  given to the Split-Merge Clustering algorithm is crippled when Split-Merge Clustering clusters the  $D_1$  segment.

Furthermore, in Table 11.10, we present the number of clusters in each data segment produced by the three algorithms. We can see that EvolveCluster produces far too few clusters when clustering  $D_1$ , with only three out of the existing seven. Similarly, PivotBiCluster is under-clustering  $D_1$  with five out of the seven clusters, partly why higher validation measures are obtained for its solution. It is only Split-Merge Clustering that manages to retain all seven clusters. These results follow the previous results on the S1 datasets, where Split-Merge Clustering consistently adapts the clustering solutions from previous segments to the new.

It is also interesting to see that all the produced clustering solutions on the Covertype dataset produce low  $SI$  scores. The two aspects that  $SI$  concerns are the compactness of the produced clusters and the separation between them. Covertype contains many features and has clusters that overlap each other in some of the features, causing  $SI$  to produce lower scores for the clustering solutions. This is evident when we compare the scores of  $F_1$ ,  $J_1$ , and  $SI$  for all three algorithms, but especially for both Split-Merge Clustering and PivotBiCluster. PivotBiCluster manages to produce an  $F_1$  score of 0.903 and a  $J_1$  score of 0.848 while only having a  $SI$  score of 0.194 (Table 11.9).

Table 11.10: Number of clusters for each algorithm on the Covertype dataset, on both the individual ( $D_0$  and  $D_1$ ) and combined ( $D_0 - D_1$ ) data segments.

Algorithm	$D_0$	$D_1$	$D_0 - D_1$
EvolveCluster	7	3	7
Split-Merge	7	7	7
PivotBiCluster	7	5	5
Ground truth	7	7	7

### 11.6.3 DELMH Dataset

The results for the DELMH dataset are presented in Tables 11.11 and 11.12. It is apparent in both tables that all three algorithms produce clustering solutions with much lower SI scores compared to the previous experiments. This is partly because of the difficulty in clustering this dataset. Most of the daily profiles in the dataset are similar to each other. During the majority of the day, there is no actual consumption of electricity. When the residents are out of their homes, only minor consumptions, such as household appliances' idle consumption, are drawn. Similarly, when the residents are asleep, only the idle consumptions are drawn.

In Table 11.12, we can see that for all segments up to  $D_2 - D_3$ , the EvolveCluster algorithm performs better in terms of the SI, but IC-av suggests there is no such clear distinction. In the final data segment, it is interesting to see that both PivotBiCluster and Split-Merge Clustering produce higher SI scores than EvolveCluster, and for Split-Merge Clustering, there is also a significantly better IC-av score. However, similarly to the results of the Covertype experiments, both SI and IC-av indicate that the produced clustering solutions are pretty poor. Based on the nature of the data, the electricity consumption of an individual household, it is natural that the produced clustering solutions are deemed poorly. Many of the daily profiles contain similar values and shapes, making it hard to distinguish between them.

### 11.6.4 Concept Drift Analysis

In this subsection, we present the results of the second experiment, where we investigate how EvolveCluster manages to handle concept drift. In Figures 11.6, 11.7, 11.8, and 11.9, we have 4 data streams that are initialized identically but have different continuations. The corresponding validation measures are presented in Tables 11.13, 11.14, 11.15, and 11.16,

Table 11.11: Results from the validation measures *Silhouette Index* and *Average Intra-Cluster Distance* on the DELMH dataset, where the all previous data segments are discarded before evaluating, for all algorithms.

		D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
EvolveCluster	SI	0.080	0.119	0.115	0.054	0.066
	IC-av	1189	672	602	570	619
Split-Merge	SI	0.080	0.105	0.150	0.077	0.116
	IC-av	1189	673	592	588	647
PivotBiCluster	SI	0.080	0.119	0.133	0.076	0.116
	IC-av	1189	661	591	595	647

Table 11.12: Results from the validation measures *Silhouette Index* and *Average Intra-Cluster Distance* on the DELMH dataset, where the previous data segment is kept during evaluationg, for all algorithms.

		D <sub>0</sub> - D <sub>1</sub>	D <sub>1</sub> - D <sub>2</sub>	D <sub>2</sub> - D <sub>3</sub>	D <sub>3</sub> - D <sub>4</sub>
EvolveCluster	SI	0.071	0.057	0.067	0.035
	IC-av	1716	1268	1107	1147
Split-Merge	SI	0.019	-0.012	0.019	0.065
	IC-av	1727	1256	1079	1054
PivotBiCluster	SI	0.002	-0.015	0.040	0.074
	IC-av	1778	1245	1080	1163

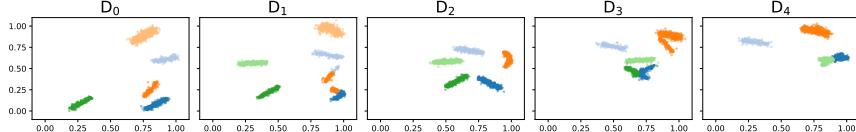


Figure 11.6: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  on the data stream generated by our RBFGenerator. Each color of the data points in the figures represents a single cluster within that segment.

respectively.

In the figures, we see constant incremental concept drift in each of the data streams. All clusters move slightly at every new data point, creating oval rather than spherical shaped clusters. We can see that EvolveCluster can

Table 11.13: Results from the validation measures  $F_1$ ,  $JII$  and  $SI$  on the data stream generated by our RBFGenerator.

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
$F_1$	1	0.854	1	0.842	0.764
Jaccard	1	0.796	1	0.760	0.662
Silhouette	0.745	0.712	0.697	0.600	0.695

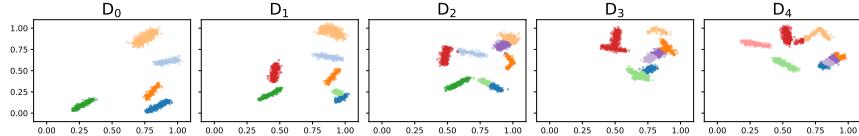


Figure 11.7: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  on the data stream generated by our RBFGenerator. Each color of the data points in the figures represents a single cluster within that segment.

Table 11.14: Results from the validation measures  $F_1$ ,  $J_1$  and  $SI$  on the data stream generated by our RBFGenerator.

	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
$F_1$	1	0.903	0.822	0.650	0.662
Jaccard	1	0.857	0.733	0.514	0.566
Silhouette	0.745	0.685	0.576	0.510	0.572

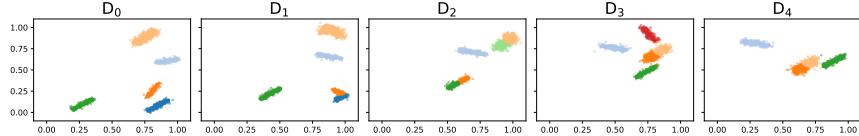


Figure 11.8: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  on the data stream generated by our RBFGenerator. Each color of the data points in the figures represents a single cluster within that segment.

Table 11.15: Results from the validation measures  $F_1$ ,  $J_1$  and  $SI$  on the data stream generated by our RBFGenerator.

	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
$F_1$	1	0.865	0.733	0.863	0.831
Jaccard	1	0.800	0.600	0.794	0.746
Silhouette	0.750	0.718	0.572	0.589	0.591

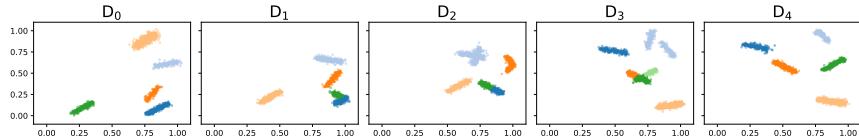


Figure 11.9: The clustering solutions obtained on each data segment  $D_0, D_1, D_2, D_3, D_4$  on the data stream generated by our RBFGenerator. Each color of the data points in the figures represents a single cluster within that segment.

Table 11.16: Results from the validation measures  $F_1$ ,  $J_1$  and  $SI$  on the data stream generated by our *RBFGenerator*.

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
$F_1$	1	0.861	0.802	0.755	1
Jaccard	1	0.799	0.703	0.643	1
Silhouette	0.749	0.640	0.664	0.657	0.802

model these incremental changes of each cluster, especially when the clusters have some distance between each other. However, when a cluster centroid reaches the boundary of the feature space, its direction is immediately changed to keep the cluster within the boundaries. For instance, in the bottom right corner of Figure 11.8 in segment  $D_1$ , we identify that the blue cluster has reached the boundary and changed its direction. EvolveCluster assigns the data points belonging to the cluster after the directional change to a new cluster as it is no longer spherical.

In each stream, we can also observe the creation or deletion of clusters in each of the segments from  $D_1$  and onwards. When new clusters appear in the stream, EvolveCluster tends to manage the addition by applying a split. However, when the clusters overlap, such as the dark blue cluster in segment  $D_2$  of Figure 11.9, the new cluster is not immediately identified.

EvolveCluster relies on the transition between segments to manage the merging of clusters, so we can notice cases of over-clustering in some segments. For instance, in segments  $D_3$  and  $D_4$  of Figure 11.7, a single cluster is divided into 3-4 separate clusters on the right side of the figures. When many clusters are close to each other, and then some of them disappear, we can observe EvolveCluster struggling to merge them. When it does not manage to merge clusters, it cascades further to not splitting the neighboring clusters if necessary accurately. This is evident in the upper right corner of the same figure, where the brown and green clusters likely should be divided into three.

## 11.7 Discussion

### 11.7.1 EvolveCluster Properties

One of the significant benefits of the proposed EvolveCluster algorithm is its simplicity in using prior knowledge about the previous data segments.

With previous centroids, EvolveCluster influences the clustering of the new segment to follow the same structure as the previous. The two aspects we have investigated using the S1 dataset seem to be handled proficiently by EvolveCluster. When we cluster the original S1 dataset, as shown in Figure 11.3, EvolveCluster discovers new clusters while maintaining the old clusters still present in the data. However, in Figure 11.3 a) under  $D_2$ , we also identify the limitations of the proposed algorithm. When two clusters are close and should be merged, it may result in an over-clustered solution. As there is no specific merge criterion for the algorithm, this is not a surprising result.

Furthermore, as shown by the results from the clustering of the continuous S1 dataset, we can see that it is easy for EvolveCluster to cluster similar behaviors that repeatedly occur in multiple segments. Contrary to both PivotBiCluster and Split-Merge Clustering, it is easy to map and identify trends that occur between segments. When PivotBiCluster and Split-Merge Clustering combine the old data segment with the new, it is clear that the relationships of the clusters between the segments are lost; thus, providing difficulties in analyzing the multiple segments and their trends.

### 11.7.2 Comparison to Other Evolving Clustering Algorithms

PivotBiCluster is continuously under-clustering the S1 datasets in this study. This can specifically be seen in Figure 11.4 and its corresponding Table 11.6. We can observe here that in the first iteration of the clustering process, PivotBiCluster reduces the number of clusters from 15 to 2 and continues to under cluster for the remainder of that experiment. This is partly why PivotBiCluster achieves a higher score on both the  $F_1$  and Jaccard measures.

With the variation of discarding the previous data segment in its produced clustering solutions, the Split-Merge Clustering algorithm seems to perform better than EvolveCluster in all the experiments in this study. Split-Merge Clustering never performs any modifications to the new clustering solution but instead combines the previous and the new solutions by morphing the old to align with the new. Since all the clustering solutions provided to Split-Merge Clustering are either based on the ground truth labels (S1 and Covertype) or extensively clustered beforehand (DELMH), the results indicate that the Split-Merge Clustering performs better than EvolveCluster. However, when we include the prior data segments and use the entirety

of the produced clustering solutions, the results significantly decreased as shown, e.g., in Figure 11.3 and Table 11.4.

### 11.7.3 Handling Concept Drift and Outliers

By analyzing the results and how EvolveCluster operates, we aim to discuss the characteristics of EvolveCluster in the presence of outliers. If a prominent outlier exists within a segment, it will not be removed directly by EvolveCluster. There are no mechanisms in place in itself that identify or removes them. Instead, if the outlier drastically differs from the other clusters, the splitting procedure of the algorithm is likely to promote the outlier to a singleton cluster. If no similar data objects arrive in future segments after the created singleton cluster, they will be discarded as a part of the clustering procedure. However, what is seen as an outlier in the current segment might not be determined to be one in future segments.

It is harder to determine if the data objects are outliers or if the streams' concepts are evolving as the stream evolves. In our second experiment, where we generated data using an RBFGenerator, we investigated how EvolveCluster models data streams with different concept drifts. We included constant incremental changes to each cluster present in the data stream, and EvolveCluster managed to model the moving clusters accurately. This type of concept drift is not always easy to identify with incremental approaches, as data is either removed or weighed down as time moves along. With EvolveCluster, we can compare the produced clustering solutions of each segment to identify if a cluster has moved significantly compared to previous segments. If more significant changes appear, it might be sufficient to compare two neighboring clustering solutions, but it is possible to compare segments further apart with drifts that appear slowly over time.

Additionally, to fully capture the dynamics of evolving data streams, we included creation and deletion of clusters in our experiments. These represent the sudden and gradual concept shift scenarios, in addition to when cluster centroids reach the boundaries and suddenly change direction. From the results, we noted that the splitting functionality of EvolveCluster generally manages to model the stream when new clusters emerge. Depending on the current status of the clustering solutions when new clusters emerge, EvolveCluster identifies the need for a split to model more accurately. However, as no specific merge criterion exists and EvolveCluster instead

relies on the transitions to merge clusters, there are occasions where both over- and under-clustering occurs. When the clustering solution produced by EvolveCluster represents a single cluster by multiple smaller clusters, it appears to disrupt the efficacy of the splitting procedure on another cluster. Specifically when the cluster to split and the ones to merge are very close to each other. These tendencies lead us to believe a merge criterion might be necessary to fully capture the dynamics of an evolving data stream, but it will add to the complexity of EvolveCluster.

## 11.8 Conclusions

This study has introduced a novel evolutionary clustering algorithm (EvolveCluster) capable of modeling data streams containing evolving data. Specifically, our experiments have shown that the proposed algorithm can retain previous trends and identify new behaviors as they emerge. Compared to similar approaches, EvolveCluster does not require each data segment to be clustered in advance, and it easily identifies the correlation of clusters between segments. Each segment in the data stream is clustered based on how the data was behaving in the previous segment, which produces an efficient clustering.

We have compared our evolutionary clustering algorithm against two similar approaches, namely the PivotBiCluster and Split-Merge Evolutionary Clustering algorithms. The results have shown that the proposed algorithm can cluster a continuous behavior over multiple data segments and identify new clusters as they emerge. Furthermore, the experiments have also revealed shortcomings of the other two algorithms where they tend to alter the given clustering solutions for the worse.

Our future work includes developing and evaluating a distributed version of the proposed evolutionary clustering algorithm. We will also investigate the possibility of a dynamic split criterion to eliminate the need for tuning a parameter. Furthermore, incorporating an additional option to remember cluster centers from data segments earlier than the previous data segment could be an exciting area of research. Adding this option could make it easier for EvolveCluster to identify recurring concepts.

## Acknowledgements

We would like to give our special thanks to Milena Angelova, one of the authors in [9], for providing us with their source code to their algorithm and experiments.

This work is funded in part by the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden.

## Appendix

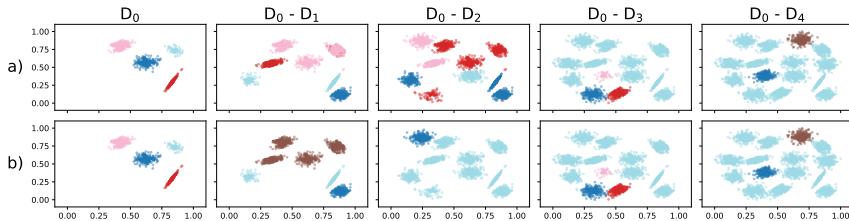


Figure 11.10: The clustering solutions obtained on each data segment  $D_0, D_1, \dots, D_4$  of the original  $S1$  dataset, where the all previous segments are included. The results generated by the three studied algorithms are depicted as follows: a) EvolveCluster, b) Split-Merge Clustering, and c) PivotBiCluster. Each color of the data points in the figures represents a single cluster within that segment.

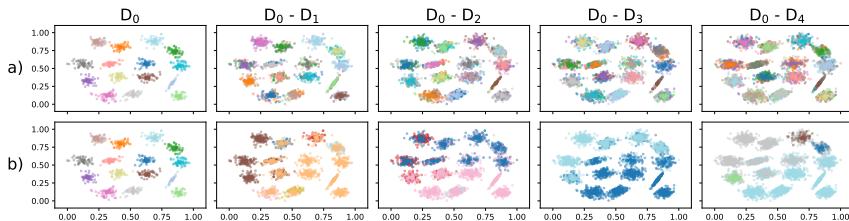


Figure 11.11: The clustering solutions obtained on each data segment  $D_0, D_1, \dots, D_4$  of the continuous  $S1$  dataset, where the all previous segments are included. The results generated by the three studied algorithms are depicted as follows: a) EvolveCluster, b) Split-Merge Clustering, and c) PivotBiCluster. Each color of the data points in the figures represents a single cluster within that segment.

## References

- [1] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. “Moa: Massive online analysis, a framework for stream classification and clustering”. In: *Proceedings of the First Workshop on Applications of Pattern Analysis*. PMLR. 2010, pp. 44–50.
- [2] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
- [3] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.
- [4] A. Zubayoglu and V. Atalay. “Data stream clustering: a review”. In: *Artif Intell Rev* 54 (2021), pp. 1201–1236.
- [5] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.
- [6] E. Lughofer. “Extensions of vector quantization for incremental clustering”. In: *Pattern recognition* 41.3 (2008), pp. 995–1011.
- [7] V. Boeva and C. Nordahl. “Modeling Evolving User Behavior via Sequential Clustering”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 12–20.
- [8] N. Ailon, N. Avigdor-Elgrabli, E. Liberty, and A. Van Zuylen. “Improved approximation algorithms for bipartite correlation clustering”. In: *SIAM J Comput* 41.5 (2012), pp. 1110–1121.
- [9] V. Boeva, M. Angelova, V. M. Devagiri, and E. Tsiporkova. “Bipartite split-merge evolutionary clustering”. In: *International conference on agents and artificial intelligence*. Springer. 2019, pp. 204–223.
- [10] P. Berkhin. “A survey of clustering data mining techniques”. In: *Grouping multidimensional data*. Springer, 2006, pp. 25–71.
- [11] D. Arthur and S. Vassilvitskii. *k-means++: The advantages of careful seeding*. Tech. rep. Stanford, 2006.
- [12] H. D. Vinod. “Integer programming and the theory of grouping”. In: *J Am Stat Assoc* 64.326 (1969), pp. 506–519.
- [13] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira. “Discussion and review on evolving data streams and concept drift adapting”. In: *Evolving systems* 9.1 (2018), pp. 1–23.

- 
- [14] K. Wadewale and S. Desai. “Survey on Method of Drift Detection and Classification for time varying data set”. In: *Int. Res. J. Eng. Technol.* 2.9 (2015), pp. 709–713.
  - [15] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. “Experimental comparison of representation methods and distance measures for time series data”. In: *Data Mining and Knowledge Discovery* 26.2 (2013), pp. 275–309.
  - [16] A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah. “A comparison study on similarity and dissimilarity measures in clustering continuous data”. In: *PloS one* 10.12 (2015), e0144059.
  - [17] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.
  - [18] C. Nordahl, V. Boeva, H. Grahn, and M. P. Netz. “Profiling of household residents’ electricity consumption behavior using clustering analysis”. In: *International Conference on Computational Science*. Springer. 2019, pp. 779–786.
  - [19] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988. ISBN: 013022278X.
  - [20] J. Handl, J. Knowles, and D. B. Kell. “Computational cluster validation in post-genomic data analysis”. In: *Bioinformatics* 21.15 (2005), pp. 3201–3212.
  - [21] C. J. Van Rijsbergen. *Information Retrieval*. 2nd. Newton, MA. 1979.
  - [22] P. Jaccard. “The distribution of the flora in the alpine zone. 1”. In: *New Phytol* 11.2 (1912), pp. 37–50.
  - [23] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. “On clustering validation techniques”. In: *Journal of intelligent information systems* 17.2-3 (2001), pp. 107–145.
  - [24] L. Vendramin, R. Campello, and E. Hruschka. “Relative clustering validity criteria: A comparative overview”. In: *Statistical Analysis and Data Mining* 3 (2010), pp. 209–235.
  - [25] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. d. Carvalho, and J. Gama. “Data stream clustering: A survey”. In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–31.

- [26] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer. “An effective evaluation measure for clustering on evolving data streams”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011, pp. 868–876.
- [27] L. E. B. Da Silva, N. M. Melton, and D. C. Wunsch. “Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study”. In: *IEEE Access* 8 (2020), pp. 22025–22047.
- [28] M. Moshtaghi, J. C. Bezdek, S. M. Erfani, C. Leckie, and J. Bailey. “Online cluster validity indices for performance monitoring of streaming data clustering”. In: *International Journal of Intelligent Systems* 34.4 (2019), pp. 541–563.
- [29] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
- [30] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.
- [31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.
- [32] M. Mousavi, A. A. Bakar, and M. Vakilian. “Data stream clustering algorithms: A review”. In: *Int J Adv Soft Comput Appl* 7.3 (2015), p. 13.
- [33] L. O’callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. “Streaming-data algorithms for high-quality clustering”. In: *Proceedings 18th International Conference on Data Engineering*. IEEE. 2002, pp. 685–694.
- [34] J. Gama, P. P. Rodrigues, and L. Lopes. “Clustering distributed sensor data streams using local processing and reduced communication”. In: *Intelligent Data Analysis* 15.1 (2011), pp. 3–28.
- [35] H.-P. Kriegel, P. Kröger, I. Ntoutsi, and A. Zimek. “Density based subspace clustering over dynamic data”. In: *International conference on scientific and statistical database management*. Springer. 2011, pp. 387–404.

- 
- [36] M. Ghesmoune, M. Lebbah, and H. Azzag. “Clustering over data streams based on growing neural gas”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2015, pp. 134–145.
  - [37] A. Zhou, F. Cao, W. Qian, and C. Jin. “Tracking clusters in evolving data streams over sliding windows”. In: *Knowledge and Information Systems* 15.2 (2008), pp. 181–214.
  - [38] S. Lühr and M. Lazarescu. “Incremental clustering of dynamic data streams using connectivity based representative points”. In: *Data Knowl Eng* 68.1 (2009), pp. 1–27.
  - [39] P. Angelov and X. Zhou. “On line learning fuzzy rule-based system structure from data streams”. In: *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 915–922.
  - [40] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving systems* 3.3 (2012), pp. 135–151.
  - [41] B. Aaron, D. E. Tamir, N. D. Rishe, and A. Kandel. “Dynamic incremental k-means clustering”. In: *2014 International Conference on Computational Science and Computational Intelligence*. Vol. 1. IEEE. 2014, pp. 308–313.
  - [42] J. P. Barddal, H. M. Gomes, and F. Enembreck. “SNCStream: A social network-based data stream clustering algorithm”. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 2015, pp. 935–940.
  - [43] J. P. Barddal, H. M. Gomes, F. Enembreck, and J.-P. Barthès. “SNC-Stream+: Extending a high quality true anytime data stream clustering algorithm”. In: *Information Systems* 62 (2016), pp. 60–73.
  - [44] Y. Mirsky, B. Shapira, L. Rokach, and Y. Elovici. “pcstream: A stream clustering algorithm for dynamically detecting and managing temporal contexts”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2015, pp. 119–133.
  - [45] F. Cao, M. Estert, W. Qian, and A. Zhou. “Density-based clustering over an evolving data stream with noise”. In: *Proceedings of the 2006 SIAM international conference on data mining*. SIAM. 2006, pp. 328–339.

- [46] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel. “Density-based projected clustering over high dimensional data streams”. In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM. 2012, pp. 987–998.
- [47] D. O. Cardoso, F. M. França, and J. Gama. “Wcds: A two-phase weightless neural system for data stream clustering”. In: *New Generation Computing* 35.4 (2017), pp. 391–416.
- [48] S. Guha and N. Mishra. “Clustering data streams”. In: *Data stream management*. Springer, 2016, pp. 169–187.
- [49] R. Anderson and Y. S. Koh. “StreamXM: An adaptive partitional clustering solution for evolving data streams”. In: *International Conference on Big Data Analytics and Knowledge Discovery*. Springer. 2015, pp. 270–282.
- [50] J. Gao, J. Li, Z. Zhang, and P.-N. Tan. “An incremental data stream clustering algorithm based on dense units detection”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2005, pp. 420–425.
- [51] N. Amit. “The bicluster graph editing problem”. PhD thesis. Citeseer, 2004.
- [52] E. Schubert and P. J. Rousseeuw. “Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms”. In: *Int Conf Similarity Search Appl*. Springer. 2019, pp. 171–187.
- [53] P. Fränti and O. Virmajoki. “Iterative shrinking method for clustering problems”. In: *Pattern Recognition* 39.5 (2006), pp. 761–765. DOI: 10.1016/j.patcog.2005.09.012.
- [54] S. Hettich and S. Bay. *The UCI KDD Archive. University of California, Department of Information and Computer Science, Irvine, CA*. 1999. URL: <http://kdd.ics.uci.edu>.
- [55] W. Toussaint. “Domestic Electrical Load Metering, Hourly Data 1994-2014. version 1.” In: <https://www.datafirst.uct.ac.za/dataportal/index.php/catalog/759> (2019).
- [56] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. “MOA: Massive Online Analysis”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 1601–1604. URL: <http://portal.acm.org/citation.cfm?id=1859903>.

- [57] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. “Scikit-multiflow: A multi-output streaming framework”. In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 2915–2914.
- [58] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. “Diagnosis of multiple cancer types by shrunken centroids of gene expression”. In: *Proceedings of the National Academy of Sciences* 99.10 (2002), pp. 6567–6572.



# MultiStream EvolveCluster

*Christian Nordahl, Veselka Boeva, and Håkan Grahn*

*In: Proceedings of the Canadian Conference on Artificial Intelligence. DOI: 10.21428/594757db.b22e0e9a*

## Abstract

This paper proposes a novel multi-stream clustering algorithm, MultiStream EvolveCluster (MS-EC), that can be used for continuous and distributed monitoring and analysis of evolving time series phenomena. It can maintain evolving clustering solutions separately for each stream/view and consensus clustering solutions reflecting evolving interrelations among the streams. Each stream behavior can be analyzed by different clustering techniques using a distance measure and data granularity that is specially selected for it. The properties of the MultiStream EvolveCluster algorithm are studied and evaluated with respect to different consensus clustering techniques, distance measures, and cluster evaluation measures in synthetic and real-world smart building datasets. Our evaluation results show a stable algorithm performance in synthetic data scenarios. In the case of real-world data, the algorithm behavior demonstrates sensitivity to the individual streams' data quality and the used consensus clustering technique.

## 12.1 Introduction

Streaming data has been growing in popularity since its inception. Today's data sources are increasingly becoming streams, partly due to the ever-growing Internet of Things (IoT) expansion. Streaming data presents challenges that are not typical in conventional datasets. Streams are faster and potentially infinite in size, which has put more conventional machine learning and data mining techniques on the sidelines. With the right type of devices, any system could be continuously monitored by small IoT devices.

To handle the enormous amount of non-labeled data sensors like these produce, algorithms have to be carefully designed to cope with these new challenges. Clustering revolves around identifying groupings of data, providing insight into the underlying structure of the data. For streaming data, clustering usually is performed with an online incremental component that uses summarization techniques to handle the tremendous amount of data, in conjunction with an offline anytime component that produces a final clustering solution.

Considering each stream as a specific view of the system, we can let each stream be modeled and analyzed individually and use the individual models to create a global model that explains the entire system using reduced communication. The communication size of sending the resulting clustering solution is much smaller than sending the raw data directly to a centralized location. This allows us both to distribute the computations and have more flexibility for the individual streams. Each stream could use its own unique combination of clustering algorithms, dissimilarity measures, and parameters. For instance, many households today contain real-time sensors for their electricity consumption. Having sensors for water, gas, outdoor and indoor climate consumption, etc., would give an overall and more complete view of the household's total consumption. The data from individual sensors could be analyzed to study and understand their specific patterns, but combining them gives an overall view of the household residents' consumption behavior.

This paper proposes a novel distributed multi-stream (multi-view) clustering algorithm entitled MultiStream EvolveCluster (MS-EC). A multi-source extension of our previously introduced algorithm EvolveCluster [1]. We design and conduct several experiments to investigate how the MS-EC algorithms perform, using synthetic and real-world data, multiple consensus clustering algorithms, and against a baseline algorithm. The obtained results show that MS-EC performs better than the baseline algorithm with respect to used evaluation measures on both synthetic and real-world data. MS-EC copes with the dimensionality curse by providing a distributed data analysis technique that generates understandable and easily interpretable results that facilitate the practitioners in monitoring and analyzing the individual data streams' behavioral patterns and the whole system's behavior.

## 12.2 Background

In smart home monitoring applications, numerous sensors capture events that occur. A conventional approach is to send all the data to a single computing device and then do data mining on the entire data gathered from all the sensors. This would require a system with considerable computational resources to focus solely on analyzing all that data. Applying data analysis separately to each stream would allow for continuous monitoring of the behavior of the individual streams and fully capture the ongoing behavior of the smart home.

The individual clustering solutions can then be sent to a centralized location for further analysis. There, we can merge them into an overall clustering solution that enables us to identify inter-cluster relationships and dependencies between the individual behaviors of the clustering solutions. This general procedure enables distributed (potentially in real-time) computing and analysis of individual sensor streams and then performing an overall analysis at a higher level of the general behavior. The individual clustering solutions would also increase the transparency of the importance of the separate streams for the overall solution, as it is easier to understand and interpret the created models with fewer data sources. Finally, it would allow different data distributions to be clustered simultaneously and separately.

The proposed MS-EC algorithm resembles *feature-distributed ensemble clustering* and *multi-view clustering* principles. Therefore, we introduce these clustering paradigms and present three types of merging procedures in ensemble clustering, commonly referred to as consensus clustering. The latter procedures are used in our experiments.

Multi-view clustering considers the data as multiple aspects of the data and its system [2]. Conventional clustering, on the other hand, only interprets the data from a single view. Dividing the data into subsets and combining certain features gives a unique system view. The views can be completely distinct or multiple views containing the same feature [3, 4]. For example, the system can have operational, contextual, and performance views, where each view has a subset of the features present in the data.

### 12.2.1 Feature-Distributed Ensemble Clustering

Ensemble learning uses multiple algorithms or models and combine their individual results to determine the end result. The idea is to use several weaker learners to create one powerful one [5]. In the case of ensemble clustering, several clustering solutions are produced and are combined to create a global clustering solution [6]. One can either use many clustering algorithms, the same algorithm with different tunings, or divide the data to create different clustering models. This paper focuses on the latter, i.e., feature-distributed ensemble learning [7]. To create the global clustering solution, consensus clustering is performed. Below, we present the three algorithms we use to produce consensus clustering in our experiments.

**Majority Vote:** An intuitive and simple consensus clustering is majority voting [8]. Each clustering solution’s cluster labels are used to vote about which pairs of objects belong together. If a threshold is reached for a pair of objects, they are paired together in the global clustering solution. To perform the majority vote, we identify the pairwise occurrences of the data objects in each clustering solution. A co-occurrence matrix,  $C_o$ , of size  $n \times n$ , where  $n$  is the number of data objects in the clustering solutions, is created and initialized to 0. For each pair of objects  $(i, j)$  in each clustering solution, we increment the corresponding cell in the matrix,  $C_o(i, j)$  with  $1/m$ , where  $m$  is the number of clustering solutions.

All pairs with co-occurrence scores larger than the threshold indicate that those two items belong in the same cluster in the consensus clustering solution. If two items are not in any cluster in the consensus clustering, a new cluster is created, and the two items are added. If one of the items belongs to a cluster and the other does not, the other item is added to that cluster. If both items belong to different clusters, those clusters are merged. Finally, if an item has no co-occurrence score over 0.5 with any other item, it is added to a singleton cluster.

**Hypergraph Partitioning:** Instead of creating a co-occurrence matrix, we can consider each clustering solution’s labels as hyperedges in a hypergraph. A hypergraph is a generalized graph where edges can connect more than two nodes, creating more of an area rather than an edge. Partitioning this hypergraph into distinct groups requires cutting edges so that no edge that overlaps groups exist. We can interpret each clustering solution’s labels as

hyperedges to create a hypergraph out of clustering solutions. Each vertex in the graph is a data instance from the clustering solutions. The vertices are connected via the hyperedges from the cluster labels [7]. This hypergraph is partitioned by cutting hyperedges that cause overlap between the groups of vertices. The resulting groups of data are the consensus clustering solution.

There are two common ways to define the stopping criterion for the partitioning procedure:  $k$ -way partitioning and  $k$ -cuts. The  $k$ -way partitioning identifies the best cuts that result in  $k$  clusters of vertices, while  $k$ -cuts partitioning identifies and performs the  $k$  cuts that provide the best distinction between the groups.

**Markov Clustering:** A middle ground between hypergraphs and co-occurrence matrices is simple graphs. To create a graph from the cluster labels, we can create a matrix by the same procedure as the one used for the co-occurrence matrix. However, each cell represents an edge in the graph. A lower score shows an edge with low weight, i.e., a closer relationship between the data objects. Any graph clustering algorithm could be used to identify the consensus clusters. In this study, we have used Markov Clustering (MC) [9], a graph clustering algorithm that performs a series of inflations and expansions on the graph matrix to cluster the vertices.

## 12.3 Related Work

With the world’s ever-increasing number of devices and sensors, we are acquiring many possible data streams. Clustering data streams has been extensively studied recently [10–13]. Most research, though, focuses on only a single data stream. In contrast, the algorithm proposed in this paper is intended for multiple streams. Multi-stream approaches can broadly be divided into two categories. The first category covers the cases where multiple streams themselves are clustered based on their characteristics [14–17]. A typical application for these is, e.g., electricity consumption, where user profiles (streams) are clustered together to identify consumer behavior. The second category comprises approaches that cluster the arriving objects themselves. These objects are either summarized (micro-clustered) on a stream level and the summaries are sent to a centralized location for a global clustering solution [18, 19], or the streams are all integrated into a single solution directly [20]. The second category focuses more on monitoring the

system's behavior and health.

The MS-EC algorithm resembles the approaches from the second category but with some caveats. Each stream represents the overall system from its perspective, and we let each stream do that by having a clustering module for each stream. The resulting clustering labels are then sent to the centralized server to perform consensus clustering. The cluster labels determine how the data points relate, seen from each stream's perspective. A procedure in line with feature-distributed clustering [7] and multi-view clustering [3]. This provides an abstract view of the system while allowing each stream to be monitored individually.

None of the previously mentioned methods focus on clustering multiple streams individually, using the resulting clustering solutions to fuse. We identified one related study, multi-view stream (MVStream) clustering [21], that approaches multi-stream clustering similarly to our proposed MS-EC algorithm. Our MS-EC algorithm is simpler, more explainable, requires less communication, and enables individual streams to be analyzed in addition to the whole system.

## 12.4 MultiStream EvolveCluster

The MS-EC algorithm consists of two separate modules, *EvolveCluster* and *ConsensusCluster*. The EvolveCluster module is an online procedure used individually by each stream to continuously maintain the partitioning of its data in clusters of similar objects. Conversely, the ConsensusCluster module is an offline procedure that can be applied at each data segment to analyze all the streams' current clustering solutions. The ConsensusCluster module is agnostic concerning the consensus clustering technique, i.e., different techniques can be implemented in this module, and the proper one can be selected especially for the task and data under consideration. We first describe these two modules and then formally describe the MS-EC algorithm's general procedure. The latter is also illustrated in Figure 12.1.

### 12.4.1 MS-EC Modules

EvolveCluster [1] is a sequential data stream clustering algorithm capable of modeling consistent and changing behaviors in a data stream. It builds upon dividing a stream into specified segments, or windows, and clusters

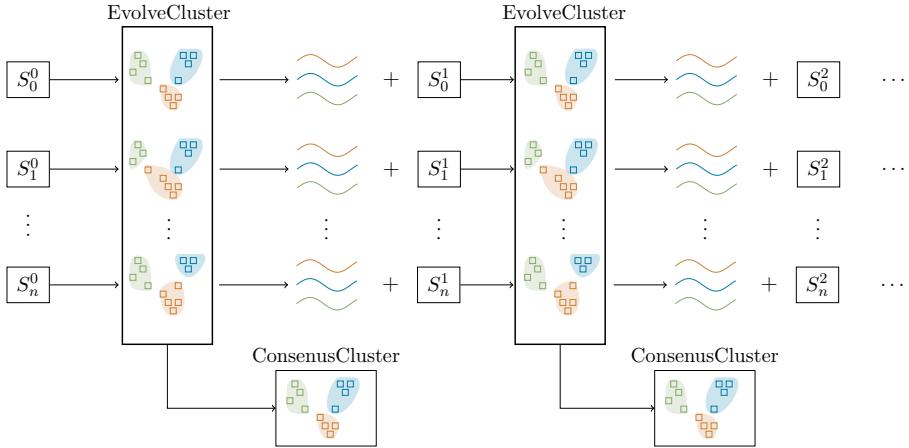


Figure 12.1: *MultiStreamEvolveClusters* general procedure while clustering a set of data streams.  $S$  is a set of  $n$  streams, i.e.  $S = \{S_0, \dots, S_n\}$ . Each stream is divided into equal-sized segments, so  $S_i^j$  is the  $j$ th segment of stream  $S_i$ . Each stream is clustered separately, and after a segment is clustered, the clusterings are combined to determine a consensus.

them sequentially as they arrive. When a segment is clustered, it is initially partitioned using prior knowledge from the previous segment. Namely, it is seeded with the cluster centroids identified in the last data segment. By incorporating this prior knowledge, the algorithm assumes the incoming data to be relatively stable. When changes occur in the stream, there is an explicit cluster-splitting criterion, based on a user-defined threshold  $\tau$ , to make the clustering solution adaptable to newly appeared concepts.

Three different variants are implemented and studied for the ConsensusCluster module of our proposed algorithm. Namely, in our experiments, we use the majority vote and two graph-based alternative algorithms (see Section 12.2). The ConsensusCluster module receives cluster labels from each stream which are then integrated by applying a consensus clustering technique selected from several implemented in the module.

## 12.4.2 MS-EC General procedure

The general working mechanism of the MultiStream EvolveCluster algorithm is illustrated in Figure 12.1. First, each data stream, or group of streams, uses its own EvolveCluster module to cluster its data. After all the streams' EvolveCluster modules have clustered a segment, the clustering solutions

are sent to the server. The server performs a consensus clustering, using a pre-selected consensus clustering technique that best fits the problem under interest. The server component is an offline procedure used to analyze and interpret the system’s behavioral patterns more abstractly. This also facilitates a better understanding of the system’s behavior by providing an overview of its performance in terms of clustering quality and the stream sensors’ performance.

Let us formally explain the two modules of MS-EC algorithm. Assume a set of streams  $S = \{S_0, S_1, S_2, \dots, S_k\}$  is given, where each stream is its unique source of information. However, each stream follows the same data collection principle and has aligned timestamps for their data. We employ a landmark window setting on our streams, creating a set of segments for each stream  $S_i = \{S_i^0, S_i^1, S_i^2, \dots, S_i^j\}$ , where  $i \in \{0, 1, \dots, k\}$  and  $j \in \{0, 1, \dots, \infty\}$ .

The corresponding sets of clustering solutions  $C = \{C_0, C_1, C_2, \dots, C_k\}$  are equally divided into segments such that  $C_i = \{C_i^0, C_i^1, C_i^2, \dots, C_i^j\}$ , where  $i \in \{0, 1, \dots, k\}$  and  $j \in \{0, 1, \dots, \infty\}$ . Each stream is individually clustered using its EvolveCluster module. When clustering stream segment  $S_i^j$ , centroids from the previous segments’ clustering solution  $C_i^{j-1}$  are used as initialization centroids. These initial centroids are removed after the initial partitioning, after which the inner clustering algorithm refines the solution with merge and split operations.

After segment  $S^j$  is clustered, creating the clustering solutions  $C_0^j, C_1^j, \dots, C_k^j$ , the cluster labels is sent to a server. This server has a ConsensusCluster module which is used to obtain a consensus clustering solution,  $CC = \{CC_0, CC_1, CC_2, \dots, CC_j\}$ , where  $j \in \{0, 1, \dots, \infty\}$ . The resulting consensus clustering solution,  $CC$ , represents the final clusters according to the ensemble knowledge gained from all the individual streams.

MS-EC’s majority of computation lies within the EvolveCluster module. The overhead added by the ConsensusCluster module is dependent on the chosen algorithm but is significantly smaller. A thorough analysis of EvolveCluster is available in our previous work [1].

## 12.5 Empirical Evaluation

This section provides the results obtained from our empirical evaluation of the proposed MS-EC algorithm. We first introduce the dissimilarity measures, evaluation measures, and software used and conclude by thoroughly explaining how our experiments have been designed.

**Dissimilarity Measures:** We investigated the use of Euclidean distance, dynamic time warping, Canberra distance, Minkowski distance, Chebyshev distance, and Mahalanobis distance by performing initial experiments on our datasets. These experiments showed that Euclidean distance performed best for the synthetic data, and for the real-world data, a mix between Euclidean distance, Minkowski distance, and Canberra distance was used.

**Evaluation Measures:** We have used multiple cluster validation indexes (CVIs) to evaluate different aspects of the clustering solutions produced by the MS-EC algorithm on the synthetic datasets. Namely, we have applied the following measures to assess the quality of the generated clustering solutions:  $F_1$  [22], Silhouette Index (SI) [23], Adjusted Rand Index (ARI) [24], and Adjusted Mutual Information Index (AMI) [25]. These measures have been used to evaluate the individual streams' clustering and consensus clustering solutions. To evaluate the clustering solutions on the real-world dataset, in addition to the SI, we apply the average intra-cluster distance (IC-av) [26]. These measures are also used to study and identify the optimal cluster number for each stream and baseline in real-world data.

**Software:** To implement the MC [9] variant of MS-EC we used the publicly available implementation<sup>1</sup> and for HGP we chose KaHyPar [27] to perform the partitioning<sup>2</sup>. The MC algorithm has one parameter that controls the cluster granularity, depicted as  $I$ , which is required for basic usage and was empirically chosen and set to 5.0 throughout our experiments. Finally, for KaHyPar we set the  $k$  parameter in  $k$ -way partitioning to the same as the maximum number of clusters available of any stream in that segment.

---

<sup>1</sup> <https://mican.org/mcl/>

<sup>2</sup> <https://kahypar.org>

### 12.5.1 Experiments

We have divided our experiments into three parts. The first experiment uses synthetic data to study the algorithm properties in a small control scenario. Namely, it exhibits a toy example of clustering 3 streams containing 2 features each. The second experiment studies the scalability of the MS-EC algorithm in a more complex scenario, again based on the synthetics data. Namely, it consists of clustering 12 8-dimensional streams. The final experiment is the clustering of the real-world dataset. The data comes from the smart building domain, consisting of individual electricity, water, and gas consumption streams with additional weather data. In this experiment, we study the algorithm sensitivity with respect to data quality in the different streams, i.e., how the degradation in performance of some stream clustering solutions affects the overall consensus clustering. All experiments' source code and results are available online<sup>3</sup>.

#### 12.5.1.1 Part 1: 3 streams of 2-dimensional data

In this experiment, we investigate and compare the different implemented variants (based on different consensus clustering techniques) of the proposed MS-EC algorithm against a baseline version of EvolveCluster that analyzes all the data gathered from all the streams together. The data streams were generated by a Radial Basis Function Generator (RBFGenerator), creating spherical clusters with a constant movement to simulate concept drift. The RBFGenerator, mentioned in [1], is based on the versions available in MOA and scikit-multiflow.

Each of the 3 streams contains 2 features in the  $[0, 1]$  range and 10,000 data points divided into 5 clusters. The baseline algorithm is fed the combined version of the three streams, a dataset consisting of 6 features and 10,000 data points. As both MS-EC and EvolveCluster algorithms rely on the initial data segment,  $S^1$ , to be clustered beforehand, we provided the ground truth labels from  $S^1$  as cluster assignments to both algorithms and the corresponding streams. The medoid for each cluster was identified by calculating the centremost point in each cluster. The  $\tau$  values for all EvolveCluster modules and the baseline were set to 0.1.

---

<sup>3</sup> <https://github.com/christiannordahl/MultiStream-EvolveCluster>

### 12.5.1.2 Part 2: 12 streams of 8-dimensional data

The streams were generated with the same RBFGenerator as in Part 1 but with 12 streams of 8 features each and 10,000 data points divided into 5 clusters. The combined dataset provided for the baseline algorithm consisted of 96 features and 10,000 data points. This experiment is intended to investigate the scalability of the algorithms and is evaluated identically to Part 1. The  $\tau$  values for all EvolveCluster modules and the baseline were set to 0.1.

### 12.5.1.3 Part 3: The Almanac of Minutely Power Dataset

The real-world dataset used in this experiment is the Almanac of Minutely Powered dataset (AMPds) version 2 [28]. It consists of minute measurements of a single residential household's electricity, water, and gas consumption in Canada from April 2012 to May 2014. Additionally, hourly weather data was collected from a nearby weather station. The streams were segmented into 12 segments, each containing 2 calendar months. They ranged from 59 to 62 days in each segment. Each stream was also normalized to the  $[0, 1]$  range.

All streams were summarized, except in the weather stream, where the mean was calculated, into hourly measurements and divided into daily profiles. We investigated the granularity of the daily profiles in the initial segment  $S^0$  by clustering 1, 2, 3, 4, 6, and 8-hour measurements, to determine an acceptable granularity for use with clustering. The final granularities chosen were 4 hours for the electricity stream ( $S_e$ ), 8 hours for the water stream ( $S_{wa}$ ), 6 hours for the gas stream ( $S_g$ ), 4 hours for the weather stream ( $S_{we}$ ), and 4 hours for the baseline.

The  $\tau$  values for each EC module was set to 0.3 for  $S_e$ , 0.37 for  $S_{wa}$ , 0.07 for  $S_g$ , and 0.1 for  $S_{we}$ . The  $\tau$  value for the baseline algorithm was set to 0.07. Euclidean distance was used for  $S_e$ ,  $S_{wa}$ ,  $S_g$ , Canberra distance for  $S_{we}$ , and Minkowski distance for the baseline.

		$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$
$S_0$	F <sub>1</sub>	1	1	1	1	0.99	0.81	0.99	1	1	1
	SI	0.84	0.89	0.90	0.85	0.77	0.58	0.78	0.80	0.81	0.83
	ARI	1	1	1	1	0.99	0.63	0.99	1	1	1
	AMI	1	1	1	1	0.99	0.77	0.99	1	1	1
$S_1$	#clust	5	5	5	5	5	5	5	5	5	5
	F <sub>1</sub>	1	1	1	1	1	1	1	1	1	1
	SI	0.90	0.88	0.82	0.83	0.87	0.86	0.80	0.76	0.80	0.76
	ARI	1	1	1	1	1	1	1	1	1	1
$S_2$	AMI	1	1	1	1	1	1	1	1	1	1
	#clust	5	5	5	5	5	5	5	5	5	5
	F <sub>1</sub>	1	0.97	0.98	0.99	0.99	0.91	1	1	1	0.99
	SI	0.59	0.73	0.76	0.75	0.73	0.67	0.75	0.83	0.87	0.73
$S_2$	ARI	1	0.98	0.98	0.99	0.99	0.84	1	1	1	0.96
	AMI	1	0.97	0.97	0.99	0.99	0.86	1	1	1	0.96
	#clust	5	5	5	5	5	5	5	5	5	5

Table 12.1: Evaluation scores generated by four CVIs, and the number of clusters, on the individual clustering solutions generated for the first 10 data segments of 3 two-dimensional streams by the EvolveCluster modules.

## 12.6 Results, Analysis, and Discussion

### 12.6.1 Part 1 results: 3 streams of 2-dimensional data

In Table 12.1, we can see that each EvolveCluster module follows suit and produces good clustering solutions overall in each stream. However, the EvolveCluster modules in stream  $S_0$  and stream  $S_2$  produce lower-quality clustering solutions for segment  $S^6$ , where we can see a drop in all CVIs. This drop occurs due to two clusters drifting on top of each other before separating away from each other again in the next segment.

In Table 12.2, we can see that when consensus clustering occurs, the above-mentioned flaw is rectified using the MV version of the MS-EC algorithm. We observe a perfect score for all segments' clustering solutions when the MV is used, but both the HGP and MC create generally lower-scoring clustering solutions. This is especially true for the HGP algorithm, the lowest-performing approach for consensus clustering in the studied context. Interestingly, the MC algorithm fluctuates in its number of clusters between the segments. The performance scores of the EvolveCluster multi-source version, which we use as a baseline model are also given in Table 12.2. It is clearly a trickier problem to cluster the combined product of the streams rather than clustering them separately first.

	$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$
MV	F <sub>1</sub>	1	1	1	1	1	1	1	1	1
	SI	0.88	0.86	0.88	0.88	0.85	0.82	0.80	0.82	0.87
	ARI	1	1	1	1	1	1	1	1	1
	AMI	1	1	1	1	1	1	1	1	1
	#clust	5	5	5	5	5	5	5	5	5
HGP	F <sub>1</sub>	0.81	0.61	0.79	0.77	0.63	0.74	0.67	0.66	0.61
	SI	0.48	0.09	0.43	0.45	0.10	0.36	0.18	0.11	0.09
	ARI	0.69	0.38	0.62	0.63	0.46	0.62	0.47	0.50	0.38
	AMI	0.79	0.51	0.72	0.74	0.63	0.74	0.60	0.62	0.52
	#clust	5	5	5	5	5	5	5	5	5
MC	F <sub>1</sub>	1	0.96	0.96	0.96	0.84	0.83	0.94	1	1
	SI	0.88	0.67	0.73	0.68	0.32	0.35	0.24	0.82	0.87
	ARI	1	0.91	0.91	0.75	0.54	0.56	0.68	1	1
	AMI	1	0.94	0.94	0.89	0.73	0.75	0.86	1	0.87
	#clust	5	4	4	4	3	3	4	5	5
Base	F <sub>1</sub>	1	0.24	0.23	0.25	0.25	0.24	0.25	0.24	0.24
	SI	0.88	-0.043	-0.047	-0.064	-0.050	-0.034	-0.063	-0.061	-0.070
	ARI	1	-0.002	-0.004	-0.002	0.002	0.002	-0.001	0.005	-0.002
	AMI	1	-0.0006	0.0008	0.0005	0.002	0.0006	0.003	0.001	0.0006
	#clust	5	5	5	5	5	5	5	5	5

Table 12.2: Comparison of the performance of the three implemented variants of the MS-EC algorithm against that of the EvolveCluster multi-source baseline version on 3 two-dimensional data streams with respect to four different CVIs and the number of clusters.

	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$	$S_{10}$	$S_{11}$
min	0.86	0.75	0.98	0.93	0.94	0.80	0.82	0.86	0.97	0.97	0.96	1
max	0.99	0.99	0.99	1	0.99	0.96	0.99	1	0.98	0.98	0.99	1
mean	0.97	0.94	0.98	0.99	0.97	0.94	0.96	0.98	0.97	0.97	0.98	1
median	0.98	0.98	0.99	1	0.95	0.96	0.99	1	0.97	0.97	0.98	1
std dev	0.04	0.081	0.0019	0.023	0.020	0.05	0.053	0.044	0.0039	0.0039	0.0084	0

Table 12.3: Statistics for the F<sub>1</sub> measure calculated for each segment on twelve 8-dimensional streams.  $S_j$  stands for the  $j$ th segment of the streams.

## 12.6.2 Part 2 results: 12 streams of 8-dimensional data

We have increased the number of streams and features for the second experiment to analyze our proposed algorithm's scalability. Table 12.3 provides the summary statistics of the F<sub>1</sub> measure on each stream on the 8-dimensional 12-stream dataset. Overall, we see a high score for all individual modules and segments. Only a few segments in streams  $S_0$ ,  $S_1$ ,  $S_5$ , and  $S_6$  produce scores lower than 0.9, with the median values staying well above 0.95.

Turning our attention to consensus clustering, we can see in Table 12.4 that the trend of more straightforward solutions performing better is also

		$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$
MV	$F_1$	1	1	1	1	1	1	1	1	1	1
	SI	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	ARI	1	1	1	1	1	1	1	1	1	1
	AMI	1	1	1	1	1	1	1	1	1	1
	#clust	5	5	5	5	5	5	5	5	5	5
HGP	$F_1$	0.58	0.71	0.71	0.71	0.70	0.71	0.70	0.70	0.71	0.71
	SI	0.15	0.26	0.22	0.22	0.19	0.15	0.19	0.18	0.26	0.23
	ARI	0.36	0.56	0.58	0.58	0.55	0.56	0.56	0.56	0.59	0.57
	AMI	0.50	0.72	0.72	0.76	0.72	0.70	0.73	0.73	0.76	0.73
	#clust	5	5	5	5	5	5	5	5	5	5
MC	$F_1$	1	0.55	0.54	0.58	0.58	0.56	0.59	0.58	0.54	0.55
	SI	0.96	-	-	-	-	-	-	-	-	-
	ARI	1	0	0	0	0	0	0	0	0	0
	AMI	1	0	0	0	0	0	0	0	0	0
	#clust	5	1	1	1	1	1	1	1	1	1
Base	$F_1$	1	0.24	0.23	0.24	0.24	0.24	0.24	0.23	0.23	0.23
	SI	0.96	-0.10	-0.097	-0.077	-0.079	-0.057	-0.12	-0.046	-0.034	-0.062
	ARI	1	-0.005	-0.002	0.008	-0.002	-0.006	-0.002	0	0	-0.004
	AMI	1	0.0008	-0.004	0.004	0.001	-0.0008	0.0004	0.003	-0.001	-0.001
	#clust	5	5	5	5	5	5	5	5	5	5

Table 12.4: Comparison of the performance of the three variants of the MS-EC algorithm against that of the EvolveCluster multi-source version on 12 eight-dimensional data streams with respect to four different CVIs and the number of clusters.

apparent here. Namely, the MV approach to consensus clustering shows a generally high score with respect to all used CVIs, with perfect scores of  $F_1$ , ARI, and AMI. The MC algorithm did not produce any good results, providing only 1 cluster for all its segments. We have investigated this further, and it turns out that when we add the 6th stream, and onwards, the produced clustering solution quickly decayed into this behavior. Both the HGP and multi-source version of EvolveCluster (the baseline) have performed similarly to what they did in the first experiment. The baseline algorithm's number of clusters has remained the same as the ground truth, but all CVIs show that the produced clustering solutions are not on par with either the MV or HGP versions of the MS-EC algorithm.

### 12.6.3 Part 3 results: AMPds2

In this experiment, we study the algorithm performance in a real-world data context. The sensitivity of the ConsensusCluster module is investigated in several experiments to reveal how the degradation in performance of some of the individual stream clustering solutions affects the overall consensus clustering.

	$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$	$S^{11}$	$S^{12}$
$S_e$	IC-av	21.8	21.6	25.1	17.5	23.7	21	21.5	18.4	17.6	23.6	24.3
	SI	0.77	0.79	0.80	0.76	0.75	0.75	0.76	0.81	0.79	0.77	0.73
	#clust	6	6	6	6	6	6	4	4	5	5	5
$S_{wa}$	IC-av	19.4	19.3	23.5	19.1	18.9	21.8	22.6	21.8	16.2	21.7	18.6
	SI	0.96	0.93	0.89	0.92	0.91	0.91	0.91	0.87	0.90	0.93	0.92
	#clust	5	5	4	3	3	4	5	5	5	5	5
$S_g$	IC-av	21.67	21.7	20.7	19.8	20.3	21.4	19.5	11.9	18.2	13.5	16.5
	SI	0.92	0.91	0.93	0.89	0.74	0.81	0.88	0.89	0.92	0.83	0.72
	#clust	5	5	4	6	6	5	4	2	3	2	2
$S_{we}$	IC-av	15.6	17.9	10.8	16.6	7.6	4.6	4.2	3.2	3.7	6.3	10.6
	SI	0.58	0.83	0.84	0.80	-	-	-	-	-	-	-
	#clust	6	3	3	2	1	1	1	1	1	1	1

Table 12.5: Evaluation scores on the individual clustering solutions generated for the first 12 data segments of 4 real-world data streams by the EvolveCluster modules.

In Table 12.5, one can see the SI and IC-av scores produced by EvolveCluster modules on the first 12 segments of the four data streams. It can be observed that the EvolveCluster has not extracted any meaningful structure for the weather stream since, from the fifth segment forward, it produces just a single cluster. This indicates that this view does not significantly impact the system behavior, and maybe it should be excluded from the data input of the ConsensusCluster module. The latter is also confirmed by the results generated by the ConsensusCluster on the integration of all four streams (see Table 12.6). Namely, we believe this affects the consensus clustering solutions produced by MV and MC. These algorithms seem to be sensitive to the quality of the clustering structure of the studied streams. A similar trend can also be noticed in the results generated by the multi-source version of EvolveCluster, the baseline.

To investigate this further, we have conducted multiple experiments studying different combinations of the four streams. In the paper, we have only included the results generated on water, electricity, and gas streams presented in Table 12.7. We can see that when excluding the weather stream, HGP’s behavior remains the same while the performance of the MV and MC variants slightly improves. An additional threshold for the majority vote, 80% beside the normal 50%, was also included, and we see an apparent increase in the number of clusters. In this scenario, many singleton clusters are created as many of the pairwise co-occurrences fall below the threshold, as now all three streams have to agree on which clusters to place the elements.

It is also interesting to notice that the trend observed in the synthetic

## 12. MULTISTREAM EVOLVECLUSTER

---

		$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$	$S^{11}$	$S^{12}$
MV	IC-av	9.2	9.5	9.3	10.9	12.8	12.2	10.8	10.1	9.1	11.3	12.7	12.0
	SI	-	-	-	-	-	-	-	-	-	-	-	-
	#clust	1	1	1	1	1	1	1	1	1	1	1	1
HGP	IC-av	9.0	9.9	9.8	8.9	12.2	10.6	9.3	8.2	9	8.6	13.3	11.5
	SI	-0.03	-0.01	-0.08	-0.06	0	-0.03	-0.07	-0.04	-0.06	-0.06	0.01	-0.06
	#clust	6	6	6	6	6	6	6	5	5	5	5	5
MC	IC-av	9.3	9.5	9.3	10.9	12.8	12.2	10.8	10.1	9.1	11.3	12.7	12.0
	SI	0.03	-	-	-	-	-	-	-	-	-	-	-
	#clust	2	1	1	1	1	1	1	1	1	1	1	1
Base	IC-av	8.9	9.3	10.3	9.4	12.5	11.5	11.1	9.2	8.7	9.7	13.1	11.9
	SI	0.15	0.16	0.10	0.19	0.23	0.14	0.14	0.11	0.15	0.20	0.25	0.24
	#clust	7	8	8	4	3	3	3	2	3	2	2	3

Table 12.6: Comparison of the performance of the three variants of the MS-EC algorithm against that of the EvolveCluster multi-source version on 4 real-world data streams.

		$S^1$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$	$S^7$	$S^8$	$S^9$	$S^{10}$	$S^{11}$	$S^{12}$
MV <sub>50</sub>	IC-av	19.3	19.2	19.4	24.2	28.8	26.6	22	21.6	18.2	24.2	27.6	25.3
	SI	-0.05	-	-	-	-	-	0.01	-	-	-	-	-
	#clust	3	1	1	1	1	1	3	1	1	1	1	1
MV <sub>80</sub>	IC-av	12.2	11.9	16	10.1	16.2	10.8	11	12.7	10.5	10.8	19.0	14.4
	SI	-0.10	-0.10	-0.17	-0.11	-0.09	-0.07	-0.11	-0.10	-0.11	-0.12	-0.10	-0.14
	#clust	28	31	26	32	29	37	30	23	23	29	22	26
HGP	IC-av	19.1	19.7	21.3	17.6	27.9	24.7	20.4	17.6	16.4	19	26.2	25.3
	SI	-0.03	-0.05	-0.06	-0.05	-0.05	-0.02	-0.08	-0.07	-0.07	-0.01	-0	-0.04
	#clust	6	6	6	6	6	6	6	5	5	5	5	5
MC	IC-av	18.7	18.7	21.3	24.2	28.8	22.3	22.7	21.6	16.5	20.4	27.6	25.6
	SI	-0.06	-0.06	-0.16	-	-	-0.11	-	-	-0.03	0.11	-	0.10
	#clust	7	7	8	1	1	11	1	1	3	4	1	2

Table 12.7: Comparison of the performance of the three variants of the MS-EC algorithm on three real-world data streams. The number of clusters produced in each segment is also depicted. The MV variant is present two times with thresholds of 50% and 80%.

data experiments is not confirmed in the real-world data. Namely, the best-performing consensus clustering technique is not the simplest, i.e., the MV, but the HPG algorithm. This once more supports that evaluating novel ML algorithms only on synthetic datasets cannot provide an objective view of the algorithms' behavior. In many cases, this is not even confirmed in a real-world data context.

One clear aspect of the MS-EC algorithm is that the ConsensusCluster module's use of only cluster labels greatly relies on the individual streams' clustering solutions. For instance, in the synthetic data experiments, we

have identified that when the individual stream clustering solutions are reasonable, the resulting consensus clustering solution tends to be of good quality. While in the real-world data experiments, the streams have been harder to cluster, leading to poor-quality clustering solutions.

The MS-EC algorithm provides a distributed clustering approach that limits communication size and keeps the data's privacy and integrity. Many stream data mining algorithms do not respect the same integrity and privacy perseverance MS-EC does; either raw data is sent directly or summary statistics to the centralized location, which then performs clustering. This is an essential aspect for the current and future computing landscapes, where cloud computing and 5G networks are still growing, to keep the privacy and integrity of the data intact.

## 12.7 Conclusions and Future Work

This paper has proposed a novel distributed multi-stream clustering algorithm, MultiStream EvolveCluster (MS-EC), with low communication. Each stream is clustered individually, and the only information sent to the centralized location is cluster labels, which preserves the privacy and integrity of the data. The foundations of the algorithm build on modularity and simplicity, where each stream can be configured precisely to its needs.

Our experiments have shown that two of the three MS-EC variants perform substantially better in clustering the synthetically made data. The third variant is also significantly better in the smaller dataset, but its performance drops when dimensionality and the number of streams increases. In the real-world data, though, we have identified that MS-EC relies heavily on the goodness and clusterability of the data in the streams. The privacy and integrity retaining setting of only submitting cluster labels is an exciting and challenging problem.

Our future plans are to dive further into the low communication and privacy setting by investigating more consensus clustering procedures. We also intend to investigate whether adding additional information to the individual streams and their clustering algorithms will positively affect the produced consensus clustering solution viability.

## Acknowledgements

We appreciate and thank Marie Netz for her support and involvement. This research was funded partly by the Knowledge Foundation, Sweden, through the Human-Centered Intelligent Realities (HINTS) Profile Project (contract 20220068).

## References

- [1] C. Nordahl, V. Boeva, H. Grahn, and M. Persson Netz. “EvolveCluster: an evolutionary clustering algorithm for streaming data”. In: *Evol. Syst.* 13.4 (2022), pp. 603–623.
- [2] S. Bickel and T. Scheffer. “Multi-view clustering.” In: *IEEE Int. Conf. Data Min. 2004*. Vol. 4. 2004, pp. 19–26.
- [3] Y. Yang and H. Wang. “Multi-view clustering: A survey”. In: *Big Data Min. Anal.* 1.2 (2018), pp. 83–107.
- [4] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang. “An overview of recent multi-view clustering”. In: *Neurocomputing* 402 (2020), pp. 148–161.
- [5] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. “A survey on ensemble learning”. In: *Front. Comput. Sci.* 14 (2020), pp. 241–258.
- [6] O. Sagi and L. Rokach. “Ensemble learning: A survey”. In: *Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery* 8.4 (2018), e1249.
- [7] A. Strehl and J. Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. In: *J. Mach. Learn. Res.* 3.Dec (2002), pp. 583–617.
- [8] A. Fred. “Finding consistent clusters in data partitions”. In: *Int. Workshop on Multiple Classifier Systems*. Springer. 2001, pp. 309–318.
- [9] S. Van Dongen. “Graph Clustering Via a Discrete Uncoupling Process”. In: *SIAM J. Matrix Anal. Appl.* 30.1 (2008), pp. 121–141.
- [10] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. d. Carvalho, and J. Gama. “Data stream clustering: A survey”. In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–31.
- [11] M. Ghesmoune, M. Lebbah, and H. Azzag. “State-of-the-art on clustering data streams”. In: *Big Data Analytics* 1.1 (2016), pp. 1–27.

- 
- [12] M. Carnein and H. Trautmann. “Optimizing data stream representation: An extensive survey on stream clustering algorithms”. In: *Business & Information Systems Engineering* 61.3 (2019), pp. 277–297.
  - [13] A. Zubaroğlu and V. Atalay. “Data stream clustering: a review”. In: *Artificial Intelligence Review* 54.2 (2021), pp. 1201–1236.
  - [14] P. Laurinec and M. Lucká. “Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting”. In: *Data Min. Knowl. Discovery* 33.2 (2019), pp. 413–445.
  - [15] K. Gajowniczek, M. Bator, and T. Ząbkowski. “Whole time series data streams clustering: dynamic profiling of the electricity consumption”. In: *Entropy* 22.12 (2020), p. 1414.
  - [16] K. Gajowniczek, M. Bator, T. Ząbkowski, A. Orłowski, and C. K. Loo. “Simulation Study on the Electricity Data Streams Time Series Clustering”. In: *Energies* 13.4 (2020), p. 924.
  - [17] A. Balzanella and R. Verde. “Histogram-based clustering of multiple data streams”. In: *Knowl. Inf. Syst.* 62.1 (2020), pp. 203–238.
  - [18] Z. Razavi Hesabi, T. Sellis, and K. Liao. “DistClusTree: A Framework for Distributed Stream Clustering”. In: *Databases Theory Appl.* Cham: Springer, 2018, pp. 288–299.
  - [19] J. S. Challa et al. “Anytime clustering of data streams while handling noise and concept drift”. In: *J. Exp. Theor. Artif. Intell.* 34.3 (2022), pp. 399–429.
  - [20] M.-H. Le-Nguyen et al. “Continuous Health Monitoring of Machinery using Online Clustering on Unlabeled Data Streams”. In: *2022 IEEE Int. Conf. on Big Data*. IEEE. 2022, pp. 1866–1873.
  - [21] L. Huang, C.-D. Wang, H.-Y. Chao, and S. Y. Philip. “MVStream: Multiview data stream clustering”. In: *IEEE Trans. Neural. Netw. Learn. Syst.* 31.9 (2019), pp. 3482–3496.
  - [22] N. Chinchor. “MUC-4 Evaluation Metrics”. In: *Proc. 4th Conf. on Message Understanding*. MUC4 '92. McLean, Virginia: Association for Computational Linguistics, 1992, pp. 22–29.

- [23] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
- [24] L. Hubert and P. Arabie. “Comparing partitions”. In: *J. Classif.* 2 (1985), pp. 193–218.
- [25] N. X. Vinh, J. Epps, and J. Bailey. “Information theoretic measures for clusterings comparison: is a correction for chance necessary?” In: *Proc. 26th Annu. Int. Conf. Mach. Learn.* 2009, pp. 1073–1080.
- [26] A. E. Baya and P. M. Granitto. “How many clusters: A validation index for arbitrary-shaped clusters”. In: *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics* 10.2 (2013), pp. 401–414.
- [27] S. Schlag. “High-Quality Hypergraph Partitioning”. PhD thesis. Karlsruhe Institute of Technology, Germany, 2020.
- [28] S. Makonin et al. “Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014”. In: *Sci. Data* 3.1 (2016), pp. 1–12.

---

# On Evaluation of Data Stream Clustering Algorithms: A Survey

---

*Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson-Netz*  
*In: blabla DOI: blabla*

## Abstract

Data stream mining is a research area that has grown enormously in the last years. The main challenge is to extract knowledge in real-time from a possibly unbounded stream of data. Clustering, a process where groupings within the data are to be identified, data streams is an interesting way to extract and identify underlying structures of the data. One of the open questions within the data stream clustering domain is regarding to its evaluation of the proposed algorithms. In this survey, we review the literature in the domain to identify the common methodologies, datasets, and evaluation measures, used to evaluate the algorithms. The literature review begins with the BIRCH algorithm from 1996 and covers the related studies published in the period up until present time, May of 2022. We identify that the methodologies primarily focus on performance and that aspects such as cluster quality are rarely visited. Performance has been the focal point of all evaluation, both in terms of computational performance and accuracy, since the inception of clustering data streams. We also identify that issues that exist in the conventional clustering domain are present in the data stream clustering. However, minor additions to the evaluation methods can improve both the applicability and usefulness of the algorithms.

## 13.1 Introduction

Over the last couple of decades, significant improvements have been reached in both computer hardware and software. Devices have become faster,

smaller, more energy-efficient, and directly influence our daily lives. Internet of Things (IoT), for instance, has achieved great prominence during this period. Cars, airplanes, smartphones, home assistance, and so on, have all become more and more intelligent due to the advances in IoT technology. An increasing number of sensors and systems continuously generate massive amounts of data. By 2020, it was estimated that each person in the world generated 1.7MB of data each second <sup>1</sup>.

In accordance with that increase of data, we also have an increasing interest in providing algorithms to manage and analyze these enormous amounts of continuous data, referred to as data streams [1, 2]. Mining such data streams involves overcoming difficulties not present in conventional data mining and knowledge discovery [3]. Data streams are potentially infinite in size, with data arriving incrementally, and, as the data is collected over time, it is likely to contain concept drift. Such difficulties require the algorithms to operate with low computational effort and to be able to adapt to the dynamic nature of the data stream [4, 5]. The majority of machine learning tasks, e.g., classification, regression, and clustering, have been investigated for this specific setting, where classification has received the most attention.

Data stream clustering algorithms have been reviewed extensively over the years [3, 6–22]. The reviews have varied between defining and extending taxonomies [3, 16], identification of algorithm and window models [3, 15, 16, 18, 20], and empirical reviews of algorithms [23–25]. Evaluation for data stream clustering has, however, remained an open question since its inception [3, 18, 24], although its importance is often stated [16–19]. A consensus has not been reached on which datasets to evaluate on, which cluster validity indices (CVIs) should be used, nor how the CVIs should be applied [14, 26].

The Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm was the first to introduce concepts that summarize data in an incremental fashion. These summaries were then used by a clustering algorithm to more efficiently cluster large data sets. The concepts of BIRCH were extended and made way to create the domain of Data Stream Clustering. Hence, in this survey, we focus on mapping how evaluation has been performed since the inception of BIRCH in 1998 up until this point in time. We will specifically look at which CVIs have been used, in what manner

---

<sup>1</sup> <http://www.domo.com/learn/data-never-sleeps-6>. Accessed Oct 2021.

they have been applied, which methodology the evaluation has followed, if and how different types of data stream clustering algorithms have been compared, and what the common evaluation datasets are. Furthermore, we will investigate specific CVIs that have been proposed for data stream clustering and the newly coined area of incremental CVIs.

Our findings show that the evaluation of data stream clustering algorithms has not received the same level of attention as developing algorithms. The domain has mainly focused on performance, where processing time, memory usage, and accuracy, are commonly used for evaluation. Qualitative aspects, such as compactness, connectedness, and separation, of the produced clusters have not been as prominent in the evaluation of data stream clustering algorithms.

## 13.2 Data Stream Clustering

Clustering in general concerns grouping similar objects together, identifying underlying concepts of datasets that distinguish the groups from each other. Each group should ideally be as compact as possible and the groups themselves should be well-separated [6]. Data stream clustering algorithms follow the same concepts as conventional clustering, but with additional obstacles to overcome.

Data streams are endless sources of data and their algorithms process the data as it arrives, either incrementally or in chunks [7]. The processing of these endless streams requires algorithms to be efficient and able to run on devices with lower computational power than usual. In the stream setting, there is also the presence of evolving data, where concepts, or behaviors, disappear, shifts, or new concepts emerge. It is therefore important to focus on the current concepts that exist in the present and put less focus on the past concepts in the stream [3]. In this section, we provide the necessary overview of data stream clustering in relation to two problem areas: *Filtering* and *Processing*.

### 13.2.1 Filtering

The evolving component of data streams requires the algorithms to continuously adapt to model the current data distribution properly. To help the algorithm focus on the present, several window models have been proposed.

These windows filter out older data and allow the algorithm to focus on the most recent data and its underlying structure. Generally, there are four types of window models in the literature [3, 16], as shown in Figure 13.1.

### 13.2.1.1 Sliding Window Models

Sliding windows are fixed-size windows that seemingly slide across the data stream. Elements enter and leave the window on a First-In-First-Out basis, retaining the most recent data objects that have arrived. Each element within the window usually has equal importance to the clustering. The size of the window often requires investigation, and no size fits neither all streams nor algorithms [7, 20]. Using a smaller window may allow the algorithm to quickly adapt to changes in the stream. On the other hand, a larger window can be more accurate for stable streams as more elements are included in the algorithm's processing.

### 13.2.1.2 Damped/Fading Window Models

In contrast to sliding window models, damped window models do not directly remove any data from their window. Instead, a decaying weight is applied to each element, where the older an element becomes, the less importance it has for the overall clustering solution. When an element gets old enough, it could be likened to being deleted as its importance to the algorithm is almost none. A typical factor for this type of window is  $2^\lambda$ , where  $\lambda$  defines the amount of decay the window performs. The decaying weight can, however, be expensive to perform every increment. To reduce the expensive computations, it is possible to update the weight within a cluster when accessed [27] or at fixed time intervals [28].

### 13.2.1.3 Landmark Window Models

As the name suggests, landmark window models have specific pre-defined landmarks where a new window starts. Each window retains the data objects until a landmark is reached. When the landmark is reached, all data objects are removed, and a new window begins. Here, data objects are usually also equally important. The landmark windows can be used in different ways. Some algorithms fully discard the previous windows after a clustering solution is produced. But some use the clustering solutions

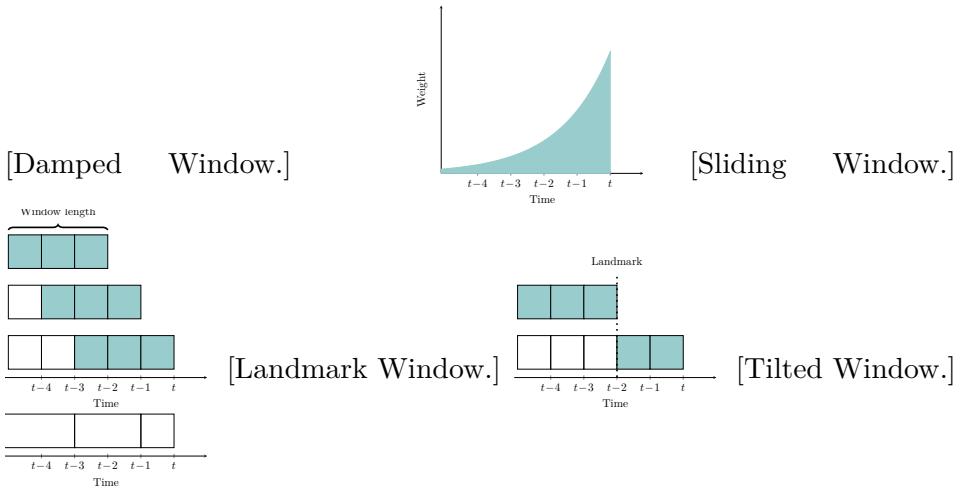


Figure 13.1: Illustration of all time window models. Adopted from Carnein and Trautmann [20] and Silva et al. [3].

from previous windows to either influence future windows or combine the clustering solutions from multiple windows into one.

We can also group algorithms here that cluster the entire stream, i.e., seeing the whole stream as a single window or the models requiring a user to manually restart the clustering when the clustering model no longer fits the stream.

#### 13.2.1.4 Tilted Window Models

The final window type, tilted window models, is somewhere between the damped/fading and the sliding window models. The most recent data is stored in a higher granularity, and the older it becomes, the less granular it is in the window [29]. In other words, the older the data becomes, the more it is summarized and averaged so it only represents a summary instead of the actual past. For instance, the last hour of data is stored in quarters, the next 24 hours are held in hourly formats, the next 31 days are stored in daily formats, etc.

#### 13.2.2 Processing

Streaming data imposes some limitations which are not present in conventional data mining. Data arrives incrementally in order, the stream size

is possibly unbounded, concepts evolve over time, and data instances are often required to be discarded after use. For clustering algorithms, this causes concerns to identify similarities and group instances together. In conventional algorithms, it is necessary that the entire dataset is present to cluster it successfully, but this is not possible for data stream clustering. Different types of models have been proposed to combat these restrictions. In Figure 13.2 we present some of the major advances in the domain.

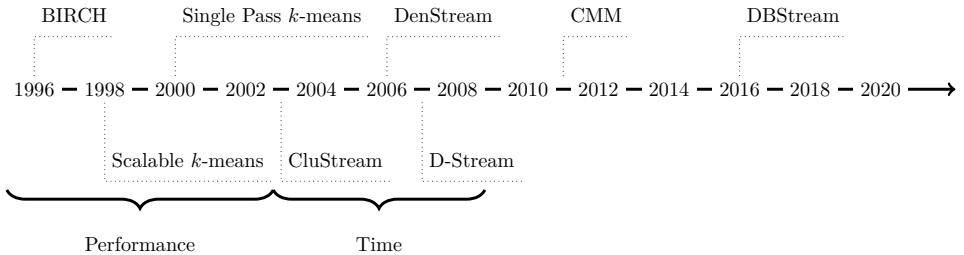


Figure 13.2: An *illustrative timeline showing the two eras we define of the data stream clustering domain. The early algorithms in the domain focused mostly on performance, reducing the computational complexities of conventional clustering algorithms to make them suitable for large databases and data streams. Then the era of time was introduced, where *CluStream* and the subsequent algorithms put focus on the current behavior of the data stream by incorporating functionality to either filter out or outweigh older data.*

One of the first algorithms applicable in this setting was BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [30]. This algorithm was intended to perform clustering over very large databases, that were impossible to contain in the main memory. To do so, BIRCH introduced two-phase clustering, where an online phase statistically summarizes the data and an offline phase produces a clustering solution based on the summary. The online phase of BIRCH processes the data incrementally in a single pass and summarizes the data elements in feature vectors which are named Clustering Feature (CF) vectors. The CF vector consists of three components:  $n$ ,  $LS$ ,  $SQ$ .  $n$  is the number of elements in the cluster,  $LS$  is a vector of dimension  $d$  of linearly summarized distances in each dimension, and  $SS$  is a vector of dimension  $d$  containing the sum of squares of all data points. These three components give the CF vector both additivity and incrementality, as all features are summarizing features, meaning both updating individual CF vectors and merging two CF vectors together is easily performed. Additionally, with these three features, it is then possible

to calculate three cluster properties to use, namely the cluster center, radius, and diameter.

BIRCH builds a height-balanced tree of these CF vectors, named a CF Tree, where each leaf node in the tree represents a cluster. When new elements are processed, BIRCH traverses the tree until a suitable CF vector that can absorb the new element, based on the CF vectors radius, is found. If no CF vector is able to absorb the element, a new CF vector is created. With the produced tree, the final CF vectors in the leaf nodes can be used as input for a conventional clustering algorithm to further refine the clusters.

### 13.2.2.1 Building on the Performance from Summarization

The CF vectors from BIRCH have been used in several adaptations of conventional clustering algorithms to make them suitable for large databases. Scalable  $k$ -means [31] and the continued single-pass  $k$ -means [32] use the same CF vectors, but are directly combined with  $k$ -means. Scalable  $k$ -means fills a buffer with the initial points of the stream and initializes in the same manner as conventional  $k$ -means. Then, the algorithm decides if it should be discarded, summarized, or retained for each point. If points are to be summarized, a CF vector is created, and the original points are discarded. The remaining points in the buffer are clustered using  $k$ -means again to create micro-clusters. These micro-clusters are clustered together using agglomerative hierarchical clustering. The newly created clusters are again summarized and replaced by a CF vector. The remaining points after these two steps are kept as individual points, and the free space in the buffer is filled with new points. Single-pass  $k$ -means operates similarly, with the only alteration that all points are discarded after the second step, thus, retaining only the  $k$  CF vectors. One of the limitations with single-pass  $k$ -means, and its predecessor, is the requirement of correctly identifying and supplying the correct  $k$ .

Of course, other summarizing methods than the CF vectors exist. Cluster centroids have also been used, albeit in a much lesser quantity [33, 34]. For instance, STREAM [33] relies on cluster centroids to represent the necessary data to cluster the data stream. STREAM divides the data stream into  $n$  chunks and clusters each chunk separately into  $k$  clusters using  $k$ -median. Whenever a final clustering is required or if the algorithm runs out of space, the  $n \times k$  cluster centroids are again clustered using the same algorithm.

The main hurdle that these algorithms aimed to solve was the performance constraints for data streams. As the entire stream would not fit in the main memory, it was essential to incrementally process the stream while still providing a good clustering solution. The temporal aspect had not yet been approached. Still, for applications requiring the entire data stream to be present in the final clustering solution, for instance, these approaches are very suitable when clustering large databases. However, when the current behaviors of the stream are of greater importance than the past behaviors, other algorithms are likely to be better suited.

### 13.2.2.2 The Temporal Aspect

With the introduction of CluStream [29] in 2003, the importance of time was incorporated into the stream clustering algorithms. CluStream built upon the groundwork that BIRCH had laid, using the two-phase clustering and the CF vector, with the addition of timestamps in the CF vector. The timestamp is used directly in the clusters to determine if a cluster has become out of date. When a cluster's average timestamp becomes lower than a threshold, that concept is no longer present in the stream and that cluster is removed. This addition allowed CluStream to produce clusterings over different time horizons, as it stored snapshots of the current micro-clusters on disk. This extension to the CF vector has been further used and developed in algorithms such as HCluStream [35], SWClustering [36], and SDStream [37].

Damped windows have, however, been the main approach for most algorithms to account for recency [27, 28, 38–64]. DenStream [28] built upon CluStream by using a damped window and defining two types of micro-clusters: potential core micro-clusters and outlier micro-clusters. A cluster is typically a potential core micro-cluster, but when the weight of a cluster is lower than a threshold, that cluster becomes an outlier micro-cluster. Then if those outlier clusters decay even further below another threshold, that micro-cluster is removed altogether. DenStream has been the basis for many subsequent algorithms, C-DenStream [58] included domain knowledge into the clustering, rDenStream [59] temporarily stores the demoted outlier micro-clusters and allows the offline process to incorporate them in the clustering to further refine the clustering, E-Stream [47] combined the damped window CF vector with histograms to determine if clusters are needed to be split or merged, etc..

Grid-based algorithms also commonly use the damped window model for their clustering. Instead of using distance-based metrics and creating micro-clusters, these approaches divide the data space into a cell grid. Data is assigned to the corresponding cells upon arrival and depending on the weight and density of each cell, the cells are deemed differently. These cells can be compared to micro-clusters, but these approaches allow the adjacent cells to be grouped directly to create macro-clusters instead of having an offline component cluster the micro-clusters. D-Stream [27] is one of the most influential grid-based approaches to data stream clustering.

Sliding window approaches appear less frequently in the literature compared to their counterparts. RepStream [65] creates a graph of nearest neighbors to create cluster representatives by incrementally inserting elements from the stream. When non-representative elements fall out of the sliding window, those are removed from the graph. DENGRIS-Stream [66] includes the sliding window in conjunction with a threshold for decaying density, such that when cells no longer exist within the window or when a cell's density is lower than the threshold, the cells are removed.

Currently, DBStream [41] is considered to be the best-performing algorithm [23]. DBStream's ingenuity lies behind keeping track of the shared density between micro-clusters, which is the number of points that lies within two microclusters, to determine if micro-clusters belong to the same macro-clusters. In an empirical review of data stream clustering algorithms, Carnein et al. [23] has shown that DBStream outperforms the other algorithms included in the review in most settings. DBStream does, however, have a higher amount of parameters to tune and is sensitive to the order elements are inserted.

### 13.2.2.3 Refraining from the Norm

A clear majority of algorithms follow the two-phase approach that was introduced with BIRCH. However, some algorithms try to follow a more conventional approach to cluster data streams. Dividing the stream into separate segments and using conventional clustering algorithms with additional functionalities is another approach. Some algorithms cluster each segment separately and then try to combine the produced clustering solutions [67]. Others let the produced clustering solutions remain separate, but use the clustering solution of previous segments to influence the next segment [68],

69].

These algorithms do not strictly follow the data stream clustering constraints because: 1) they move away from the incremental nature of the domain and 2) they do not necessarily allow the user to gather the current clustering model at any time. However, this type of clustering can be beneficial in certain applications where the data stream is not as rapid and there are fewer requirements for producing the offline clustering.

### 13.3 Evaluating Clustering Algorithms

Evaluating the performance of algorithms has many different aspects, including computational performance, memory usage, correctness, etc.. To evaluate the clusters themselves, both in terms of quality and correctness, we have two major categories of CVIs: *internal* and *external* [70]. Internal CVIs focus on the clustering solution itself, using only the data and the produced clusters to determine how well the clustering solution is. They evaluate aspects such as the compactness, separation, and connectedness, of the produced clusters [71]. Conversely, external CVIs either compare against the ground truth or another clustering solution to determine how accurate the produced clustering solution is in comparison. Over the years, a large number of CVIs and methodologies have been proposed in the conventional clustering domain [72].

#### 13.3.1 Cluster Evaluation in Data Streams

Specifically for data stream clustering, less research has been focused on producing CVIs and methodologies [3]. However, the importance of the evaluation is often stated [3, 16–19]. Kokate et al. [13] brings up a few CVIs to use within data stream clustering but misses to report the common trends and what methodologies are used.

##### 13.3.1.1 Cluster Validity Indices

The literature on data stream clustering shows that many common approaches from conventional clustering have been of key importance when evaluating in the domain. In Table 13.1, we present the most commonly used CVIs. The full list is available in Table .5. It is clear that external CVIs are much more common compared to internal CVIs.

Table 13.1: *Cluster validity indices used within the data stream clustering area. Full table available in the Appendix, Table .5.*

Type	CVI	Article
External	Average cluster purity	[13, 23, 28, 35, 37–39, 41, 46, 47, 49, 52, 54, 56, 57, 59, 60, 63–65, 73–88]
	Weighted average purity	[24, 50, 61, 89]
	Accuracy	[44, 45, 55, 78, 79, 90–93]
	Adjusted Rand Index	[13, 23, 41, 50, 94–97]
	F-measure	[47, 50, 62, 63, 67, 68]
Internal	Cluster Mapping Measure	[24, 80, 98, 99]
	SSQ	[13, 24, 27, 29, 32, 33, 41–43, 62, 77, 89, 100–103]
	Loglikelihood	[31, 36, 103, 104]
	Silhouette	[13, 67, 68, 105]

**External CVIs** One of the most emergent trends among the external CVIs for data stream clustering is the importance of accuracy. The majority of algorithms are evaluated using measures such as purity, average cluster purity, and accuracy, as shown in Table 13.1. These measures focus fully on the correctly classified instances and disregard the quality of the clusters.

Purity is a simple and transparent measure, which focuses fully on the accuracy of the produced clustering solution. Its simplicity is, however, also its drawback. Purity produces no penalty for the number of clusters in the resulting clustering solution, which could lead to misleading results. Additionally, it makes comparison hard between different clustering solutions as the results are not truly comparable unless both clustering solutions contain the same number of clusters.

The purity of a single cluster is calculated as the ratio of correctly assigned elements within a cluster, giving:

$$\text{purity} = \frac{|C_i^d|}{|C_i|}, \quad (13.1)$$

where  $|C_i^d|$  is the number of points in a cluster belonging to the majority class and  $|C_i|$  the number of points in the true cluster. Both the average and weighted average cluster purity is used. The average purity is defined as:

$$\text{purity} = \frac{1}{k} \sum_{i=1}^k \frac{|C_i^d|}{|C_i|}. \quad (13.2)$$

The weighted average purity, commonly denoted average cluster purity

in the literature, is computed as:

$$\text{purity} = \sum_{i=1}^k \frac{|C_i|}{n} \cdot \frac{|C_i^d|}{|C_i|}, \quad (13.3)$$

which can be simplified to:

$$\text{purity} = \frac{1}{n} \sum_{i=1}^k |C_i^d|, \quad (13.4)$$

Accuracy, similarly to purity, fully focuses on the correctly classified instances in a cluster. It calculates the ratio of the correctly classified instances, which can be defined as:

$$\text{Accuracy} = \frac{\sum_{i=1}^k |C_i^d|}{\sum_{i=1}^k |C_i|}, \quad (13.5)$$

where  $|C_i^d|$  is the number of instances in a cluster belonging to the majority class and  $|C_i|$  is the number of instances in the true cluster.

Adjusted Rand Index (ARI) is a variant of the Rand Index that is corrected for chance. ARI measures the similarity between two clustering solutions and is commonly used to compare against the ground truth clustering. Using a contingency table, the overlap between the clustering solutions  $X$  and  $Y$  is summarized by calculating the number of instances  $n_{ij}$  in common between  $X_i$  and  $Y_j$ .

	$Y_0$	$Y_1$	$\dots$	$Y_j$	$sums$
$X_1$	$n_{11}$	$n_{12}$	$\dots$	$n_{1j}$	$a_1$
$X_2$	$n_{21}$	$n_{22}$	$\dots$	$n_{2j}$	$a_2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$X_i$	$n_{i1}$	$n_{i2}$	$\dots$	$n_{ij}$	$a_i$
$sums$	$b_1$	$b_2$	$\dots$	$b_j$	

The sums  $b_1, b_2, \dots, b_j$  and  $a_1, a_2, \dots, a_i$  are calculated as the sum of the number of instances in common, for instance  $b_1 = \sum_{k=0}^i n_{k1}$ . The sums  $a_{ij}$  and  $b_{ij}$  are used in conjunction with the number of instances in common  $n_{ij}$  to calculate the ARI as follows:

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}. \quad (13.6)$$

The  $F_1$  measure is a commonly used metric in conventional cluster analysis and it also captures the accuracy of a clustering solution. It is calculated as the harmonic mean of the precision and recall of the produced clustering solution compared to the ground truth. It can be defined as:

$$F_1(C_i) = \frac{2|C_i \cap C_i^d|}{|C_i| + |C_i^d|}, \quad (13.7)$$

where  $C_i$  is the cluster containing the maximum number of objects from  $C_i^d$ .

To evaluate the overall  $F_1$  score for the clustering solution  $C$ , two common approaches are used, micro and macro average. The macro average sees all classes as equal while the micro average corrects the score by each individual class's frequency. The macro  $F_1$  is defined as:

$$F_1(C) = \frac{1}{l} \sum_{i=1}^l F(C_i). \quad (13.8)$$

The only CVI that is specifically designed for data stream clustering, to our knowledge, is the Cluster Mapping Measure (CMM) [106]. It is designed to consider the special properties of data stream mining, which includes

- Aging of data instances
- Missed points
- Misplaced points
- Noise

A penalty function  $pen$  is used in combination with a calculation of connectivity  $con$  and the weight  $\omega$  of an instance to calculate the  $CMM$ . It is defined as

$$CMM(C, CL) = 1 - \frac{\sum_{o \in \mathcal{F}} \omega(o) \cdot pen(o, C)}{\sum_{o \in \mathcal{F}} \omega(o) \cdot con(o, CL)}, \quad (13.9)$$

where  $\omega$  is the weight of an instance  $o$ ,  $pen(o, C)$  is the penalty of an instance  $o$  in the clustering solution  $C$ , and  $con(o, Cl)$  the connectivity of the instance  $o$  to cluster  $Cl$ . The full definition and explanation can be found in their paper for the interested reader.

**Internal CVIs** Interval CVIs are not as common to apply for evaluation compared to external CVIs. The most common internal CVIs are the sum of squared distances (SSQ), loglikelihood, and the Silhouette Index (SI). SSQ is the only internal CVI employed in more than 4 studies.

SSQ calculates the sum of squared distances within a cluster, which corresponds to identifying how tight a cluster is, and it should be minimized. It is defined as:

$$\text{SSQ} = \sum_{i=1}^k \sum_{j=1}^m d(c_i, c_{ij}) \quad (13.10)$$

where  $k$  is the number of clusters,  $m$  is the number of data points in a cluster, and  $c_{ij}$  is the data object  $j$  in cluster  $c_i$ .

SI also measures how tight a cluster is, but in combination with how the is between the clusters in the clustering solution. It is defined as:

$$\text{SI}(C) = \frac{1}{n} \sum_{i=1}^n \frac{(b_i - a_i)}{\max\{a_i, b_i\}} \quad (13.11)$$

where  $n$  is the number of elements,  $a_i$  is the mean distance of instance  $i$  to every other instance in its cluster  $C_j$ , and  $b_i$  the smallest mean distance of instance  $i$  to all points in any other cluster  $C_k, k \neq j$ .

### 13.3.1.2 Incremental CVIs

Incremental cluster validity indices (iCVIs) are a fairly new concept that emerged with the publication of the incremental Xie-Beni index (iXB) [107]. Many traditional CVIs, such as SI, Xie-Beni index, Davies-Bouldin index, etc., have then been converted to be applicable in an incremental scenario [107–109].

iCVIs differ from their original counterparts in the sense that the iCVIs are on a more abstract level. They are not as precise in their judgment of a clustering solution. For instance, the incremental version of SI calculates its score based on the cluster centroids, instead of each individual element in each cluster as in the original SI. Using the centroids instead of each element tends to produce more optimistic scores compared to its original counterpart.

Table 13.2: *Other evaluations performed in the studies.*

Type	Evaluation Measure	Article
Computational	run time	[27, 28, 30, 32, 33, 36, 39, 41–43, 45–53, 55, 57, 63–65, 75–77, 80, 82–85, 87, 88, 91–93, 98–102, 110–112]
	Memory usage	[28, 32, 36, 41, 43, 44, 46, 49, 50, 53, 55, 56, 59, 64, 77, 78, 85, 86, 90, 91, 113]
	Update time	[44, 46, 78, 81, 86, 97]
	Points per second	[54, 61–63, 73, 74]
Distance	RAM-hours	[98, 99]
	Distance to true cluster centre	[31, 104]
Graphical	Visualization	[13, 30, 93, 111, 112, 114–116]

### 13.3.2 Other Aspects to Evaluate

The additional requirements of real-time processing and low computational costs for the devices that should run the data stream clustering algorithms is clearly shown in the evaluation of the algorithms. Much focus is put on the run time, memory usage, and processing time per data point. In Table 13.2, we summarize the most commonly used evaluation measures, and we provide the full contents in Table .6.

## 13.4 Evaluation Methodologies

### 13.4.1 Evaluation of Two-Phase Algorithms

One of the difficulties with data stream clustering, and data stream mining in general, is the importance of recency. The stream itself is dynamic, the behavior of its data changes over time. Usually, we want to put focus on the current behavior of the stream because we are interested in what the data currently looks like. We only want to allow the historical behaviors to slightly influence the current modeling. The recency aspect is also present in the evaluation methodologies of data stream clustering algorithms. The window approaches described in Section 13.2.1 are commonly used to view the algorithms' ability in producing good clusters in the streaming data setting.

In the initial stages of data stream clustering, there were already some algorithms present that were aimed at the clustering of large databases that would not fit in main memory, such as BIRCH [30] and Scalable K-means [31]. These algorithms were not intended for application within a data

Table 13.3: *Evaluation methodologies used.*

Evaluation Method	Article
Window Evaluation	[13, 24, 27–29, 33, 35, 37, 38, 45–54, 56–60, 62–65, 74–78, 80–83, 86, 88, 90, 92, 96, 97, 103]
Entire Datasets	[30–32, 40, 43, 53, 55, 61, 79, 85, 87, 89, 91, 95, 98, 99, 101, 102, 104, 111–116, 118–123]
Prequential Evaluation	[23, 41, 42]

stream mining setting and, thus, did not incorporate mechanics to adapt the clustering as time progresses. It was not until the inception of CluStream [29] that the aspect of time was included. Not only in the algorithm’s definition, but also in its evaluation.

Another concern with the evaluation of two-phase algorithms is the creation of micro-clusters to summarize the data in the stream. When two-phase algorithms are evaluated, the assignment of each element from the stream to the corresponding micro-cluster is not available. That is the point of the online component of the algorithms, summarizing the data stream to reduce the memory requirements instead of using the original data. Thus, to evaluate an algorithm’s performance on a stream, the mapping between original data points to corresponding micro-clusters must be kept separately when evaluating the algorithm. This can, however, also be done afterward by assigning the data point from the stream to the closest micro-cluster [23, 117].

#### 13.4.1.1 Evaluation within a Window

A common approach to evaluate two-phase algorithms is to use CVIs within a window, as seen in Table 13.3. The algorithms are allowed to use the entire stream to their disposal, thus, letting algorithms that employ a faded window to work correctly. However, only the elements within the window are used in the calculation of the CVIs. The first occurrence of this is in [29], where the authors use purity within the window.

The window size is defined the same way as a landmark and sliding, with the additional inclusion of the number of horizons (i.e. windows) to include. For instance, a window size of 1’000 ( $w = 1000$ ), and a horizon of 2 ( $h = 2$ ) would include 2’000 elements in each increment. Window size and the number of horizons differ between studies, depending on what the authors aim to show. A larger window of the stream shows how the algorithm performs with long-term behaviors, while a smaller window showcases the

algorithm's capabilities of quick adaptations [16]. It is more closely related to a landmark window approach, but as the horizons does not tend to be very large it also bears resemblance to a sliding window.

There are also cases where a batch processing is performed which we also group in this evaluation type. These windows, however, tend to be much larger (for instance 20% of the stream in a single window). Additionally, it is also more common for the incremental and online algorithms to use these window sizes, or batch evaluation.

#### 13.4.1.2 Prequential Evaluation

Another type of used evaluation method, although not as common, is prequential evaluation, or Interleaved Test-Then-Train, where an algorithm is “trained” on a partition of the stream and then the obtained clustering solution is used to “classify” the coming  $x$  items [16]. These  $x$  elements are included in the training set for the next increment, and the next  $x$  items are used for testing.

This method can be seen as an extension of the landmark window evaluation method, which incorporates the evaluation of unseen data. The quality of the produced clustering is not affected by the size of the window, only the evaluation itself will be affected. Using this method, Carnein et al. [23] evaluated both the macro- and micro-clusters of several algorithms.

#### 13.4.2 Evaluation of Other Algorithm Types

For algorithms that do not follow the two-phase approach, the evaluation tends to follow the evaluation of more conventional clustering. Incremental algorithms [124] that operate on the data directly have the original data objects in their clustering solution. Depending on the type of evaluation that is performed, there are examples of both evaluation at specific landmarks and a single calculation at the end. Evaluating at specific landmarks performs the CVI calculations on the entire stream up until that point in time. Showing how the algorithm performs as it progresses over the stream.

Algorithms that divide a stream into fixed-sized chunks [67, 68, 125], also evaluate similarly to conventional clustering. Depending on how the chunks are treated, their evaluation also differs. If each chunk is viewed individually in the clustering, each chunk is usually evaluated separately as

well. But, if the chunks are combined to create a larger clustering solution, the chunks are also evaluated when combined.

### 13.4.3 Efficiency and Complexities

Efficiency, both in terms of computational run time and energy consumption, is an important aspect of data stream mining in general. Algorithms are supposed to operate on devices with low computational power and possibly on battery-powered devices. There has always been a priority on creating algorithms that can process the data stream quickly. Thus, it is no surprise that a major part of the evaluation of these algorithms has revolved around the computational effort to run them.

run time, i.e. the elapsed time to process a stream from start to finish, has been the most prominent measure that algorithms have been evaluated with, with memory usage being the second. A summarized view of the evaluation measures used can be found in Table 13.2. However, there have also been some algorithms that have been evaluated with the time it takes for a single item to be processed (update time) and how many points per second the algorithm can process. The update time and points per second show another dimension on top of just the run time, as we can see how the processing for a single item relates to the current model of the algorithms. When more of the stream is processed, the larger the clustering model becomes in the algorithm. This size increase of the model also increases the computational efforts for each individual point. Some authors include the analysis of the individual elements of the stream over time and show how it scales in relation to the number of clusters, and so on, in the model.

### 13.4.4 Datasets

Being able to evaluate algorithms on datasets containing concept drift is a tricky subject. In real-world datasets, it is hard to know if concept drift exists but more specifically when they occur. There exist some in the public domain, and a few have been used extensively within the domain. On the other hand, the synthetic datasets can easily contain evolving concepts and it is known beforehand where they exist. However, it might not be accurate to real-world scenarios. In Table 13.4 we show the most commonly used real-world datasets and summarize which studies have used self-made synthetic and publicly available synthetic datasets.

Table 13.4: Datasets used to evaluate data stream clustering algorithms.

Type	Datasets	Article
Real World	KDD-CUP'99	[23, 27–29, 33, 35, 37–39, 41–43, 45, 46, 49, 50, 52–54, 56–60, 63–65, 73–78, 80–84, 86–89, 92, 95–100, 120, 123]
	Covertype	[23, 24, 41, 42, 48, 52–54, 56, 61, 62, 65, 67, 68, 73, 75, 81–83, 86, 87, 96–99, 101, 102, 122, 126]
	KDD-CUP'98	[28, 29, 32, 58, 77, 98, 99]
	Tower	[101, 102, 122]
	Census 1990	[101, 102, 122]
	Iris	[34, 79, 118]
	Physiological Data Modeling Contest	[57, 62, 80, 114, 115, 121]
	Sensor <sup>2</sup>	[23, 41, 42]
Synthetic	Electricity	[24, 98, 99]
	Selfmade synthetic	[13, 23, 27–33, 36, 38, 39, 41, 44, 46–50, 53–55, 57, 58, 61, 62, 64, 65, 68, 73, 75–78, 85, 86, 89–91, 93, 94, 97–100, 102–105, 110–116, 118, 119, 121, 126]

#### 13.4.4.1 Real World Datasets

The two most common real-world data sets that are used for the evaluation of data stream clustering algorithms are the KDD'CUP-99 dataset, or a subset of it, and the Covertype dataset. The KDD-CUP'99 dataset consists of raw TCP-dump data of network traffic over a period of 9 weeks. It is divided into a training set and a test set, which have approximately 4'900'000 and 2'000'00 records, respectively. A subset of the data set is also used quite commonly in the domain, where approximately 490'000 records are used.

One of the interesting aspects of the KDD-CUP'99 data set is that the training and testing set are drawn from different probability distributions. In the training set, there are 24 different attacks that occur in the network, while in the testing set there are an additional 14 attacks that have not been seen before. Merging these data sets together gives a clear location in the stream where new concepts will start to emerge.

Covertype is a dataset of cartographic variables describing which type of forest is present. It consists of 581'012 objects with 54 attributes divided among 7 classes. This dataset is mainly used due to its size, a larger amount of features, and uneven class distribution. The data set is in reality not a data stream, but when used in this domain the data set is viewed as a continuous stream with a constant flow of data.

#### 13.4.4.2 Synthetic Datasets

Synthetic data sets are commonly used and are often needed to be able to run experiments for the algorithms. One group of synthetic sets is the data sets containing non-spherical clusters. These are used to showcase an algorithm's ability to detect arbitrary-shaped clusters, usually when the algorithms consist of either density-based or grid-based clustering.

Another is to use a generator to create a synthetic stream. Here, it is possible to exactly identify at which point in time certain scenarios occur which allows for a thorough analysis of an algorithm's capabilities. For instance, incorporating clusters that emerge, disappear, merge together, or splits off into two clusters, gives the dynamic behavior of a data stream that an algorithm should be able to adapt to. Radial Basis Function (RBF) generators are commonly used and are implemented in many of the available platforms for data stream mining, such as Massive Online Analysis (MOA)<sup>3</sup> and RiverML<sup>4</sup>. This type of generator allows you to incorporate a constant drift of the cluster centers, such that the data that is generated will always be slightly changing.

Many static synthetic data sets that have been used in conventional clustering are also included in the evaluation of data stream clustering algorithms.

### 13.5 Discussion

#### 13.5.1 Evaluation Methods

The premise of putting emphasis on the current behavior of the data stream is prominent also in its evaluation. Algorithms in this domain should be able to adapt as the stream progresses and try to forget the non-recurring behaviors of the past. Thus, when evaluating data stream algorithms, the approaches tend to put focus on the current data by filtering out the old by, for instance, using a window or prequential evaluation.

Changing the size of the window and the number of horizons will capture different aspects. Having a larger number of data points when calculating CVIs will show how stable the algorithm's produced clusters are. In contrast,

---

<sup>3</sup> <https://moa.cms.waikato.ac.nz>

<sup>4</sup> <https://riverml.xyz/>

a smaller amount of data points will show more how quickly and accurately the algorithms adapts to changes. The size of the window is important and requires some attention from the researcher, similarly to the use of windows directly in algorithms.

One thing that differs prequential evaluation methodology compared to window evaluation, is when clusters disappear or emerge in the window to be tested. If a new cluster emerges during that window, the clustering solution would not have seen those elements before and would cause a lower score compared to if it was evaluated with the window approach. This might not be so misleading, however, as it would be evident that a new concept has emerged and if the algorithm successfully adapts for the next window it shows its capabilities.

### 13.5.2 Correctness Over Cluster Quality

One of the concerns that we have identified is the lacking evaluation of cluster quality. The evaluation often defaults to calculating purity in an evaluation window, which requires a dataset with a ground truth label or a comparing clustering solution. There is no real indication of how well the clusters are formed. Speed and performance have been a pillar of necessity in this domain and the correctness of the clustered elements has gone hand in hand with this evaluation.

The single most used CVI within the domain is the average cluster purity, normally in combination with the window evaluation. This combination is so commonly used it can be seen as the de facto evaluation method for data stream clustering algorithms, at least the two-phase algorithms. No other CVI is used to this extent, even though there are no obstacles to using any other CVI within the window. It follows the indication that correctness is deemed much more important than cluster quality. Many CVIs would, however, be applicable for use in this evaluation method. Particularly, internal CVIs could be of direct interest.

Purity, in general, should not be used to compare solutions with a different number of clusters. To increase the purity score of a clustering solution, a simple way is just to increase the number of clusters and it is likely to rise. This is partly why it has been remedied by using the average cluster purity. However, it does not really affect the scoring that much as all algorithms tend to achieve a 0.99 purity when the micro clusters are

evaluated. The micro clusters represent only a few elements each, and there is an abundant number of micro-clusters. This, in itself, will produce a high score of purity. Albeit it is important to evaluate the micro clusters, there should also be more effort included to evaluate the macro clusters and how they perform.

To use the internal CVIs, you would need to produce a macro-clustering solution at each increment as they require the entire macro-clustering solution to judge the quality. Additionally, all the data included in the current window would have to be correctly assigned and included in the computation of the CVI. Arguably, the requirements of that additional computations could cause experiments to be time-consuming. However, here the use of iCVIs might be a possible contender. These measures incrementally update themselves, such that the scores do not have to be recomputed after each element that arrives in the stream.

The iCVIs will, however, not be able to judge the quality as well as the original CVIs, such as the incremental Silhouette Index compared to the traditional Silhouette Index, as they are commonly a bit too generous in their scoring. But, they will give a good approximation of the quality of the produced macro-clusters. Furthermore, using these iCVIs within the algorithm themselves, if they contain split/merge mechanisms, would allow many algorithms to gather multiple inputs from the iCVIs to base their split/merge decisions.

One issue that still remains is the neglect of the offline component in most algorithms. Most, if not all, the focus is put on the online component of the algorithms that summarizes the stream by creating micro-clusters. There is rarely an evaluation of the offline component when the algorithms are from the same family. The argument behind that seems to be that both algorithms would have access to the same offline algorithm. Thus, the algorithms that produce better micro-clusters would likely be better when aiding the offline algorithms to create the macro-clusters.

### **13.5.3 Comparing Different Families of Algorithms**

Proposed algorithms in the area are usually evaluated against algorithms in the same family. For instance, algorithms that have been developed as extensions of the framework of CluStream are commonly evaluated against CluStream and other algorithms that have been developed as extensions of

the mentioned algorithm. This is by no means a bad evaluation, but it can be seen as quite one-dimensional.

It is important to get a view of how an algorithm performs against the general competition in the entire domain. Some of the studied algorithms do perform tests against more than the algorithms in their family, such as [29], but the general trend is to remain in the family. One of the, if not the, most contributing study for a general comparison of data stream clustering algorithms is performed by Carnein et al. [23]. In the study, the authors compare 10 of the most prolific algorithms in the data stream clustering domain. Using prequential evaluation, the authors do remain neutral in their comparison as all algorithms are allowed to use the entire history of the data stream if they desire. However, for algorithms that are rapidly adapting, this type of evaluation might not cater as much to their preference. Furthermore, dividing the stream into batches causes concern about when the division occurs and how large each chunk is.

It is hard to identify a single methodology that provides the fairest comparison. Combining multiple evaluation methodologies provides a more thorough investigation of the algorithm, but also increases the complexity. It might, however, be worth the additional complexities.

#### 13.5.4 Can we use iCVIs?

iCVIs suit algorithms such as StreamKM, where the entire stream is clustered in the end. Since the iCVIs do not remove the old data, such as a sliding window, the current concepts present in the cluster solutions are not premiered. But, including the removal of old elements, would that make it harder to compute? Would it be unreasonable to keep all data points in the window to evaluate? Will it differ from just using a window to evaluate?

The positive effect of these metrics is that they are suitable for use in the incremental clustering of streams. Especially in those data streams where the entire stream is important, as the CVIs do not take recency into consideration. None of the proposed iCVIs remove any of the data that the algorithm no longer puts importance to. This could cause the measures to after a while show bad values for each algorithm. The authors in [107] mention that it is not efficient to keep a vector of all the elements while the evaluation occurs, which is true when the entire stream is taken into consideration. But, when seen in a window evaluation method it can be

an interesting way forward for these measures. Additionally, and maybe also primarily, they could be used as performance indicators in the online algorithms and to detect changes in the data, i.e. a change detector.

In their current state, the iCVIs are mostly suitable for fully incremental and online algorithms. Two-phase algorithms operate differently, as they create an abundance of micro-clusters in their online phase and rely on their offline algorithm to create the macro-clusters. Furthermore, the common approach in the two-phase algorithms is to employ some kind of window model to filter out older data, which is not applicable to the current iCVIs.

### 13.5.5 Data is Always Needed

A common approach to evaluating a data stream clustering algorithm is to combine the KDD-CUP'99 and Covertype datasets in conjunction with some synthetic datasets that target the algorithm's specific use case. Although Covertype has been used extensively in the data stream clustering domain, it is not a streaming data set in reality. Its initial introduction to the evaluation of data stream clustering algorithms was due to it being a real-world data set that was larger than the norm.

One of the advantages of the KDD-CUP'99 dataset is that the training and testing set are not from the same probability distribution. Only 24 types of attack are present in the training set, while those 24 plus an additional 14 are present in the testing set. This drastic change exhibits those concept drift scenarios that are commonly included in the data stream setting. It is, however, unclear in many cases if both sets are used or if only the training set is used for evaluation.

Though, KDD-CUP'99 also contains some troubling concerns. The distribution of data within the dataset is not the most promising for evaluating data stream clustering algorithms. Some parts of the dataset contain only a single type of attack. For instance, when KDD-CUP'99 is divided into subsets containing 10% of the data each, the smurf attack is the only present type of instance in subsets 4-7, and the 8th subset consists almost entirely of Neptune attacks [127]. For some segments of the data stream to only contain essentially one cluster could showcase the algorithms' capabilities of adapting to the stream. However, as mentioned in [23], when only a single type of cluster exists for a larger period of time, no algorithm will really have an issue in producing a high-scoring clustering solution. Causing inflation of

the scoring for some algorithms.

It is clear from the literature that additional data sets from the real world are needed. Although it is admirable that an entire domain has managed to all adopt these two real-world data sets in their experiments, it is clear that they contain issues that might provoke better results from algorithms compared to the real world. Gathering these data sets is not easy, and a lack of real-world data sets is a common problem in data mining. It is, however, necessary to further improve the area of data stream clustering.

Synthetic data sets are commonly used to show an algorithm's capabilities in clustering non-spherical clusters. Typically, all density-based and grid-based algorithms are partly evaluated with these data sets just to show they are capable of clustering non-spherical clusters. Many algorithms are also evaluated on their capabilities of adapting to concept drift scenarios, where clusters emerge, disappear, merge together, and split. These scenarios are harder to pinpoint where they occur in real-world data. Adding these generated data sets to the evaluation, where the occurrences of the concepts drift are known, can aid in showcasing the capabilities of algorithms. Both in the actual development and evaluation of an algorithm.

### 13.5.6 Going Forward

The initial question we asked ourselves when writing this survey, was if we could identify a sound methodology for evaluating data stream clustering algorithms. Venturing through the domain literature we were able to identify a common methodology that is used throughout, but it is, however, not ideal. Compared to other domains of clustering, the majority of the evaluation revolves around performance, both in terms of correctness and computational efforts.

We had the belief that this could not be the only aspect of the algorithms that was of interest to neither the authors nor the domain itself. Even though the main characteristics of the domain has revolved around performance since its beginning, due to the constraints of data streams, there must be an interest of also evaluating the other aspects of clustering as well. Over time we affirmed our beliefs that there is no “one way to rule ‘em all” and a similar article made the affirmation even stronger, namely “Clustering: Science or Art?” by Luxburg et al. [128].

Even though we narrow the domain down significantly by adhering to

the data stream concepts of incrementality and speed, there is a vast sea of applications these types of algorithms can be used for. For instance, network traffic, stock exchange, meteorological sensors, sales records in stores, website user patterns, etc. These are all application areas that may share some similarities and definitely contain differences. Evaluating algorithms in these different areas requires different methodologies and evaluation measures.

Therefore, we lean to agree with Luxburg et al. [128] regarding a need to create a more systematic approach to the evaluation of clustering algorithms. Creating a catalog of data stream clustering problems, that both contain the data and the recommended evaluation process, is an interesting and fruitful idea. It will, however, be a time consuming task.

In the meantime, it is both easy and meaningful to add additional aspects when algorithms are evaluated. Changing some focus from the micro-cluster and the online-phase of the two-phase algorithms towards the macro-clusters themselves would give more meaningful evaluations. Firstly, it will be easier to compare algorithms within the entire domain. Some algorithms do not lean on the two-phase approach and have no direct comparison against the micro-clusters. Having actual results from the offline component of the two-phase algorithms would make evaluation and comparison easier in general. Secondly, evaluating the macro-clusters more often would also open up for other CVIs to be used than those currently in use. Internal CVIs that assess the quality of the produced clusters would give an additional aspect rather than simply focusing on the correctness and performance that are currently evaluated. Lastly, it would also make it more accessible to practitioners in the domain as these results will give an indication of how the algorithm performs in practice.

## 13.6 Conclusions

Evaluating data stream clustering algorithms is still an open research issue. In this review, we have identified the most common methodologies for evaluation and also identified what needs to be improved. First and foremost, the evaluation of algorithms puts full focus on correctness and has no aspect of quality. Average purity is still the most common metric used for evaluating the clustering capabilities of algorithms, even though other CVIs could just as easily be calculated. However, there is no single choice of CVIs that can be regarded as the best. There is a multitude of application areas and many

of them do not view the aspects of clustering in the same way. Depending on what the intended application of the algorithm is, e.g., data pre-processing to improve other tasks or exploratory analysis, there are many different aspects to look at [128].

Second, like in the rest of the data mining community, there is a need for more real-world data. Currently, most algorithms in the domain are evaluated on the KDD-CUP'99 and Covertype datasets, which both have issues in their structure. Datasets are usually hard to come by and data stream clustering is no exception. There are, however, more and more opportunities for datasets as we have an ever-increasing number of connected devices in the world.

Finally, although many promising algorithms have been developed over the years, the general methodology for the evaluation of them has not received as much effort, which was identified already back in 2013 by Silva et al. [3]. The general procedure remains the same, which causes hindrance to real-world application, in an otherwise very exciting and promising domain. It would be ideal to create a catalogue of clustering problems, both their data and recommended evaluation, to have general methodologies for evaluation. This catalogue will, however, require lots of effort and will not be created in the near future. In the meantime, adding additional CVIs to current methodologies will help to give additional aspects of the clusters to evaluate.

## Declarations

### Funding

Not applicable.

### Conflict of interest/Competing interests

Not applicable.

### Ethics approval

Not applicable.

Consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

Data gathered from the literature review are available in their raw format as appendices.

Code availability

Not applicable.

Authors' contributions

Christian Nordahl presented the idea for the paper, performed the literature search and the main part of the data analysis, and wrote the initial draft. All authors participated in the data analysis and results discussions, as well as edited and commented on previous manuscript versions. All authors read and approved the final manuscript.

mybibfile

## .1 CVIs and Evaluation Measures Used in the Domain.

Table .5: *Cluster validity indices used within the data stream clustering area.*

Type	CVI	Article
External	Average cluster purity	[13, 23, 28, 35, 37–39, 41, 46, 47, 49, 52, 54, 56, 57, 59, 60, 63–65, 73–88]
	Weighted average purity	[24, 50, 61, 89]
	Accuracy	[44, 45, 55, 78, 79, 90–93]
	Adjusted Rand Index	[13, 23, 41, 50, 94–97]
	F-measure	[47, 50, 62, 63, 67, 68]
	Cluster Mapping Measure	[24, 80, 98, 99]
	Normalized Mutual Information	[38, 49, 50, 87, 96]
	Rand Index	[50, 53, 58, 87]
	Entropy	[65, 85, 96]
	Modified Hubert's $\tau$	[114, 115]
	V-measure	[96, 119]
	Error rate	[34, 88]
	Precision	[51, 62]
	Recall	[51, 62]
	Jaccard Index	[50, 67, 68]
	Recognition rate	[118, 119]
	Detection rate of new clusters	[59]
	Folkes and Mallow Index	[50]
Internal	SRAND	[122]
	# of micro clusters	[13]
	SSQ	[13, 24, 27, 29, 32, 33, 41–43, 62, 77, 89, 100–103]
	Loglikelihood	[31, 36, 103, 104]
	# of clusters	[13, 78, 96, 105]
	Silhouette	[13, 67, 68, 105]
	Dunn's Validity Index	[114, 115]
	Weighted Entropy	[31, 104]
	SSE ?	[79, 85]
	Simplified Silhouette	[97]
	Davies-Bouldin Index	[79]
	Root Mean Square (RMS)	[123]
	cRand	[13]

## .1. CVIs and Evaluation Measures Used in the Domain.

---

*Table .6: Other evaluation performed in the studies.*

Type	Evaluation Measure	Article
Computational	Run time	[27, 28, 30, 32, 33, 36, 39, 41–43, 45–53, 55, 57, 63–65, 75–77, 80, 82–85, 87, 88, 91–93, 98–102, 110–112]
	Memory usage	[28, 32, 36, 41, 43, 44, 46, 49, 50, 53, 55, 56, 59, 64, 77, 78, 85, 86, 90, 91, 113]
	Update time	[44, 46, 78, 81, 86, 97]
	Points per second	[54, 61–63, 73, 74]
	RAM-hours	[98, 99]
	# of distance computations	[99]
Distance	Distance to true cluster centre	[31, 104]
	Median/Max radius	[61]
	Loss to true cluster centre	[121]
	Pairwise intra cluster distance	[30]
Detection?	Correct proportion of clustering	[120]
	Detection accuracy	[120]
	False alarm rate	[120]
	False negative rate	[120]
Graphical	Visualization	[13, 30, 93, 111, 112, 114–116]
Misc.	Reachability distance	[126]
	Ordering	[126]
	Number of clusters	[13]
	No Measures	[116]

## 2 Datasets Used in the Domain.

Table .7: *Most commonly used datasets to evaluate data stream clustering algorithms.*

Type	Datasets	Article
Real World	KDD-CUP'99	[23, 27–29, 33, 35, 37–39, 41–43, 45, 46, 49, 50, 52–54, 56–60, 63–65, 73–78, 80–84, 86–89, 92, 95–100, 120, 123]
	Covertype	[23, 24, 41, 42, 48, 52–54, 56, 61, 62, 65, 67, 68, 73, 75, 81–83, 86, 87, 96–99, 101, 102, 122, 126]
	KDD-CUP'98	[28, 29, 32, 58, 77, 98, 99]
	Physiological Data Modeling Contest	[57, 62, 80, 114, 115, 121]
	Tower	[101, 102, 122]
	Census 1990	[101, 102, 122]
	Iris	[34, 79, 118]
	Sensor <sup>5</sup>	[23, 41, 42]
Synthetic	Electricity	[24, 98, 99]
	Self-made synthetic	[13, 23, 27–33, 36, 38, 39, 41, 44, 46–50, 53–55, 57, 58, 61, 62, 64, 65, 68, 73, 75–78, 85, 86, 89–91, 93, 94, 97–100, 102–105, 110–116, 118, 119, 121, 126]

## .2. Datasets Used in the Domain.

---

Table .8: *Less common datasets used for evaluating data stream clustering algorithms.*

Datasets	Article
Airlines	[98, 99]
Reuters Information Retrieval	[31, 104]
REV Digits Recognition	[31, 104]
Wine	[67, 79]
Electrical Demand dataset	[114, 115]
Yeast	[67]
Anthropometric	[67]
Spambase	[102]
Intrusion	[102]
Bavarian Newborn Screening	[110]
NIST	[118]
Pendigit	[119]
Optdigit	[119]
Mnist	[119]
Documentary <sup>6</sup>	[119]
DS1 <sup>7</sup>	[87]
Letter4 <sup>8</sup>	[87]
Sea <sup>9</sup>	[87]
Hyperplan	[87]
EGEE	[88]
SEQUOIA	[111]
Geolife [129]	[111]
NFD	[36]
CalTech128	[101]
Bloomed	[34]
Breast	[34]
Ionosphere	[34]
Mushroom	[34]
Leukemia	[34]
Lymphoma	[34]
Balance	[34]
Statlog (shuttle) dataset	[113]
Abaole	[113]
Powersupply <sup>10</sup>	[42]
Japan Weather Data	[51]
Landsat Sattelite data	[50]
Stock Market data	[94]
Human Activity Recognition (HAR)	[95]
Hear0 Smartphone Sensor data	[95]
Body Posture and Movement (BPoM)	[99]
NID	[96]
Gas Sensors [130]	[96]
Traffic data from Aarhus <sup>11</sup>	[105]
Person Activity [131]	[97]
Seeds	[79]
Glass	[79]
Breast Cancer	[79]
IIP <sup>12</sup>	[86]
Adult dataset	[24]
Poker Hand	[24]
DSD_ Benchmark <sup>13</sup>	[13]
Individual Household Electric Power Consumption <sup>14</sup>	[13]
DELMH	[68]
S1 & S2	[68]

### .3 Raw Data Tables from Literature Review.

## References

- [1] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu. “Data stream analysis: Foundations, major tasks and tools”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.3 (2021), e1405.
- [2] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, et al. “Open challenges for data stream mining research”. In: *ACM SIGKDD explorations newsletter* 16.1 (2014), pp. 1–10.
- [3] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. d. Carvalho, and J. Gama. “Data stream clustering: A survey”. In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–31.
- [4] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
- [5] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. “Moa: Massive online analysis, a framework for stream classification and clustering”. In: *Proceedings of the First Workshop on Applications of Pattern Analysis*. PMLR. 2010, pp. 44–50.
- [6] J. Gama and M. M. Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.
- [7] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*. 2nd ed. SIAM, 2020.
- [8] H. K. Singh and V. Katiyar. “Data Stream Partitioning Algorithms for Big Data Analytics: A Review”. In: *Presented at the 3rd Global Leadership Research Conference*. 2018.
- [9] A. Amini and T. Y. Wah. “Density micro-clustering algorithms on data streams: A review”. In: *World Congress on Engineering 2012. July 4-6, 2012. London, UK*. Vol. 2188. International Association of Engineers. 2010, pp. 410–414.
- [10] M. Mousavi, A. A. Bakar, and M. Vakilian. “Data stream clustering algorithms: A review”. In: *Int J Adv Soft Comput Appl* 7.3 (2015), p. 13.

- 
- [11] Y. Yang, Z. Liu, and Z. Xing. “A review of uncertain data stream clustering algorithms”. In: *2015 Eighth International Conference on Internet Computing for Science and Engineering (ICICSE)*. IEEE. 2015, pp. 111–116.
  - [12] S. Ding, F. Wu, J. Qian, H. Jia, and F. Jin. “Research on data stream clustering algorithms”. In: *Artificial Intelligence Review* 43.4 (2015), pp. 593–600.
  - [13] U. Kokate, A. Deshpande, P. Mahalle, and P. Patil. “Data stream clustering techniques, applications, and models: comparative analysis and discussion”. In: *Big Data and Cognitive Computing* 2.4 (2018), p. 32.
  - [14] A. Zubaroğlu and V. Atalay. “Data stream clustering: a review”. In: *Artificial Intelligence Review* 54.2 (2021), pp. 1201–1236.
  - [15] D. Toshniwal et al. “Clustering techniques for streaming data-a survey”. In: *2013 3rd IEEE international advance computing conference (IACC)*. IEEE. 2013, pp. 951–956.
  - [16] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng. “A survey on data stream clustering and classification”. In: *Knowledge and information systems* 45.3 (2015), pp. 535–569.
  - [17] M. Ghesmoune, M. Lebbah, and H. Azzag. “State-of-the-art on clustering data streams”. In: *Big Data Analytics* 1.1 (2016), pp. 1–27.
  - [18] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues. “Clustering algorithms: A comparative approach”. In: *PloS one* 14.1 (2019), e0210236.
  - [19] D. Xu and Y. Tian. “A comprehensive survey of clustering algorithms”. In: *Annals of Data Science* 2.2 (2015), pp. 165–193.
  - [20] M. Carnein and H. Trautmann. “Optimizing data stream representation: An extensive survey on stream clustering algorithms”. In: *Business & Information Systems Engineering* 61.3 (2019), pp. 277–297.
  - [21] A. Zubaroğlu and V. Atalay. “Data Stream Clustering: A Review”. In: (2020).
  - [22] C. C. Aggarwal. *Data streams: models and algorithms*. Vol. 31. Springer Science & Business Media, 2007.

- [23] M. Carnein, D. Assenmacher, and H. Trautmann. “An empirical comparison of stream clustering algorithms”. In: *Proceedings of the computing frontiers conference*. 2017, pp. 361–366.
- [24] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis. “An evaluation of data stream clustering algorithms”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 11.4 (2018), pp. 167–187.
- [25] A. R. Mahdiraji. “Clustering data stream: A survey of algorithms”. In: *International Journal of Knowledge-based and Intelligent Engineering Systems* 13.2 (2009), pp. 39–44.
- [26] J. Gama, P. P. Rodrigues, and R. Sebastiao. “Evaluating algorithms that learn from data streams”. In: *Proceedings of the 2009 ACM symposium on Applied Computing*. 2009, pp. 1496–1500.
- [27] Y. Chen and L. Tu. “Density-based clustering for real-time stream data”. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 133–142.
- [28] F. Cao, M. Estert, W. Qian, and A. Zhou. “Density-based clustering over an evolving data stream with noise”. In: *Proceedings of the 2006 SIAM international conference on data mining*. SIAM. 2006, pp. 328–339.
- [29] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang. “A framework for clustering evolving data streams”. In: *Proceedings 2003 VLDB conference*. Elsevier. 2003, pp. 81–92.
- [30] T. Zhang, R. Ramakrishnan, and M. Livny. “BIRCH: A new data clustering algorithm and its applications”. In: *Data Mining and Knowledge Discovery* 1.2 (1997), pp. 141–182.
- [31] P. S. Bradley, U. Fayyad, and C. Reina. “Scaling Clustering Algorithms to Large Databases”. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. KDD’98. New York, NY: AAAI Press, 1998, pp. 9–15.
- [32] F. Farnstrom, J. Lewis, and C. Elkan. “Scalability for clustering algorithms revisited”. In: *ACM SIGKDD Explorations Newsletter* 2.1 (2000), pp. 51–57.

- 
- [33] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. "Streaming-data algorithms for high-quality clustering". In: *Proceedings 18th International Conference on Data Engineering*. IEEE. 2002, pp. 685–694.
  - [34] E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. Gama. "Olindda: A cluster-based approach for detecting novelty and concept drift in data streams". In: *Proceedings of the 2007 ACM symposium on Applied computing*. 2007, pp. 448–452.
  - [35] C. Yang and J. Zhou. "Hclustream: A novel approach for clustering evolving heterogeneous data stream". In: *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*. IEEE. 2006, pp. 682–688.
  - [36] A. Zhou, F. Cao, Y. Yan, C. Sha, and X. He. "Distributed data stream clustering: A fast em-based approach". In: *2007 IEEE 23rd Int. Conf. Data Eng.* IEEE. 2007, pp. 736–745.
  - [37] J. Ren and R. Ma. "Density-based data streams clustering over sliding windows". In: *2009 Sixth international conference on fuzzy systems and knowledge discovery*. Vol. 5. IEEE. 2009, pp. 248–252.
  - [38] A. Forestiero, C. Pizzuti, and G. Spezzano. "A single pass algorithm for clustering evolving data streams based on swarm intelligence". In: *Data Mining and Knowledge Discovery* 26.1 (2013), pp. 1–26.
  - [39] A. Amini, T. Y. Wah, et al. "Leaden-stream: A leader density-based clustering algorithm over evolving data stream". In: *Journal of Computer and communications* 1.05 (2013), p. 26.
  - [40] M. Ghesmoune, H. Azzag, and M. Lebbah. "G-stream: Growing neural gas over data stream". In: *International Conference on Neural Information Processing*. Springer. 2014, pp. 207–214.
  - [41] M. Hahsler and M. Bolaños. "Clustering data streams based on shared density between micro-clusters". In: *IEEE Transactions on Knowledge and Data Engineering* 28.6 (2016), pp. 1449–1461.
  - [42] M. Carnein and H. Trautmann. "evostream—evolutionary stream clustering utilizing idle times". In: *Big data research* 14 (2018), pp. 101–111.

- [43] J. Gao, J. Li, Z. Zhang, and P.-N. Tan. “An incremental data stream clustering algorithm based on dense units detection”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2005, pp. 420–425.
- [44] N. H. Park and W. S. Lee. “Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams”. In: *Data & Knowledge Engineering* 63.2 (2007), pp. 528–549.
- [45] C. Jia, C. Tan, and A. Yong. “A grid and density-based clustering algorithm for processing data stream”. In: *2008 Second International Conference on Genetic and Evolutionary Computing*. IEEE. 2008, pp. 517–521.
- [46] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang. “Density-based clustering of data streams at multiple resolutions”. In: *ACM Transactions on Knowledge discovery from Data (TKDD)* 3.3 (2009), pp. 1–28.
- [47] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai. “E-stream: Evolution-based technique for stream clustering”. In: *International conference on advanced data mining and applications*. Springer. 2007, pp. 605–615.
- [48] Y. Yang, Z. Liu, J.-p. Zhang, and J. Yang. “Dynamic density-based clustering algorithm over uncertain data streams”. In: *2012 9th international conference on fuzzy systems and knowledge discovery*. IEEE. 2012, pp. 2664–2670.
- [49] A. Amini, H. Saboohi, T. Ying Wah, and T. Herawan. “A fast density-based clustering algorithm for real-time internet of things stream”. In: *The Scientific World Journal* 2014 (2014).
- [50] A. Amini, H. Saboohi, T. Herawan, and T. Y. Wah. “MuDi-Stream: A multi density clustering algorithm for evolving data stream”. In: *Journal of Network and Computer Applications* 59 (2016), pp. 370–385.
- [51] M. Motoyoshi, T. Miura, and I. Shioya. “Clustering stream data by regression analysis”. In: *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation- Volume 32*. Citeseer. 2004, pp. 115–120.

- 
- [52] D. K. Tasoulis, N. M. Adams, and D. J. Hand. “Unsupervised clustering in streaming data”. In: *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*. IEEE. 2006, pp. 638–642.
  - [53] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng. “SVStream: A support vector-based algorithm for clustering data streams”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.6 (2011), pp. 1410–1424.
  - [54] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. “A framework for projected clustering of high dimensional data streams”. In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 2004, pp. 852–863.
  - [55] N. H. Park and W. S. Lee. “Grid-based subspace clustering over data streams”. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 2007, pp. 801–810.
  - [56] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel. “Density-based projected clustering over high dimensional data streams”. In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM. 2012, pp. 987–998.
  - [57] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl. “Density-based projected clustering of data streams”. In: *International Conference on Scalable Uncertainty Management*. Springer. 2012, pp. 311–324.
  - [58] C. Ruiz, E. Menasalvas, and M. Spiliopoulou. “C-denstream: Using domain knowledge on a data stream”. In: *International Conference on Discovery Science*. Springer. 2009, pp. 287–301.
  - [59] L.-x. Liu, Y.-f. Guo, J. Kang, and H. Huang. “A three-step clustering algorithm over an evolving data stream”. In: *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*. Vol. 1. IEEE. 2009, pp. 160–164.
  - [60] J. Lin and H. Lin. “A density-based clustering over evolving heterogeneous data stream”. In: *2009 ISECS international colloquium on computing, communication, control, and management*. Vol. 4. IEEE. 2009, pp. 275–277.

- [61] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. “The clustree: indexing micro-clusters for anytime stream mining”. In: *Knowledge and information systems* 29.2 (2011), pp. 249–272.
- [62] M. Hassani, P. Kranen, and T. Seidl. “Precise anytime clustering of noisy sensor data with logarithmic complexity”. In: *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*. 2011, pp. 52–60.
- [63] W. Meesuksabai, T. Kangkachit, and K. Waiyamai. “Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty”. In: *International Conference on Advanced Data Mining and Applications*. Springer. 2011, pp. 27–40.
- [64] C. Isaksson, M. H. Dunham, and M. Hahsler. “SOStream: Self organizing density-based clustering over data stream”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2012, pp. 264–278.
- [65] S. Lühr and M. Lazarescu. “Incremental clustering of dynamic data streams using connectivity based representative points”. In: *Data Knowl Eng* 68.1 (2009), pp. 1–27.
- [66] A. Amini, T. Y. Wah, and Y. W. Teh. “DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window”. In: *Proc. international conference on data mining and computer engineering*. 2012, pp. 206–210.
- [67] V. Boeva, M. Angelova, V. M. Devagiri, and E. Tsiporkova. “Bipartite split-merge evolutionary clustering”. In: *International conference on agents and artificial intelligence*. Springer. 2019, pp. 204–223.
- [68] C. Nordahl, V. Boeva, H. Grahn, and M. Persson Netz. “EvolveCluster: an evolutionary clustering algorithm for streaming data”. In: *Evolving Systems* (2021), pp. 1–21.
- [69] C. Åleskog, V. M. Devagiri, and V. Boeva. “A Graph-Based Multi-view Clustering Approach for Continuous Pattern Mining”. In: *Recent Advancements in Multi-View Data Analytics*. Ed. by W. Pedrycz and S.-M. Chen. Cham: Springer International Publishing, 2022, pp. 201–237. ISBN: 978-3-030-95239-6. DOI: 10.1007/978-3-030-95239-6\_8.
- [70] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988. ISBN: 013022278X.

- 
- [71] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. “On clustering validation techniques”. In: *Journal of intelligent information systems* 17.2-3 (2001), pp. 107–145.
  - [72] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. “An extensive comparative study of cluster validity indices”. In: *Pattern Recognition* 46.1 (2013), pp. 243–256.
  - [73] C. C. Aggarwal and S. Y. Philip. “A framework for clustering uncertain data streams”. In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 150–159.
  - [74] Y. Lu, Y. Sun, G. Xu, and G. Liu. “A grid-based clustering algorithm for high-dimensional data streams”. In: *International Conference on Advanced Data Mining and Applications*. Springer. 2005, pp. 824–831.
  - [75] Y. Sun and Y. Lu. “A grid-based subspace clustering algorithm for high-dimensional data streams”. In: *International Conference on Web Information Systems Engineering*. Springer. 2006, pp. 37–48.
  - [76] A. Namadchian and G. Esfandani. “DSCLU: a new Data Stream CLUstring algorithm for multi density environments”. In: *2012 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing*. IEEE. 2012, pp. 83–88.
  - [77] A. Zhou, F. Cao, W. Qian, and C. Jin. “Tracking clusters in evolving data streams over sliding windows”. In: *Knowledge and Information Systems* 15.2 (2008), pp. 181–214.
  - [78] R. Hyde, P. Angelov, and A. R. MacKenzie. “Fully online clustering of evolving data streams into arbitrarily shaped clusters”. In: *Information Sciences* 382 (2017), pp. 96–114.
  - [79] K.-s. Zhang, L. Zhong, L. Tian, X.-y. Zhang, and L. Li. “DBIECM—an Evolving Clustering Method for Streaming Data Clustering”. In: *Amse Journals-Amse Iieta* 60.1 (2017), pp. 239–254.
  - [80] M. Hassani, P. Spaus, A. Cuzzocrea, and T. Seidl. “Adaptive stream clustering using incremental graph maintenance”. In: *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. PMLR. 2015, pp. 49–64.

- [81] V. Bhatnagar and S. Kaur. “Exclusive and complete clustering of streams”. In: *International Conference on Database and Expert Systems Applications*. Springer. 2007, pp. 629–638.
- [82] G. Huang, D. Liang, J. Ren, and C. Hu. “An algorithm for clustering uncertain data streams over sliding windows”. In: *6th International Conference on Digital Content, Multimedia Technology and its Applications*. IEEE. 2010, pp. 173–177.
- [83] G.-Y. Huang, D.-P. Liang, C.-Z. Hu, and J.-D. Ren. “An algorithm for clustering heterogeneous data streams with uncertainty”. In: *2010 International Conference on Machine Learning and Cybernetics*. Vol. 4. IEEE. 2010, pp. 2059–2064.
- [84] H. Hentech, M. S. Gouider, and A. Farhat. “Clustering heterogeneous data streams with uncertainty over sliding window”. In: *International Conference on Model and Data Engineering*. Springer. 2013, pp. 162–175.
- [85] A. Al Abd Alazeez, S. Jassim, and H. Du. “EDDS: An enhanced density-based method for clustering data streams”. In: *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*. IEEE. 2017, pp. 103–112.
- [86] C. Jin, J. X. Yu, A. Zhou, and F. Cao. “Efficient clustering of uncertain data streams”. In: *Knowledge and Information Systems* 40.3 (2014), pp. 509–539.
- [87] M. Ghesmoune, M. Lebbah, and H. Azzag. “Clustering over data streams based on growing neural gas”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2015, pp. 134–145.
- [88] X. Zhang, C. Furtlechner, and M. Sebag. “Data streaming with affinity propagation”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2008, pp. 628–643.
- [89] R. Anderson and Y. S. Koh. “StreamXM: An adaptive partitional clustering solution for evolving data streams”. In: *International Conference on Big Data Analytics and Knowledge Discovery*. Springer. 2015, pp. 270–282.
- [90] N. H. Park and W. S. Lee. “Statistical grid-based clustering over data streams”. In: *ACM Sigmod Record* 33.1 (2004), pp. 32–37.

- 
- [91] D. Barbará and P. Chen. “Using the fractal dimension to cluster datasets”. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 260–264.
  - [92] L. Wang and H. Li. “Clustering algorithm based on grid and density for data stream”. In: *AIP Conference Proceedings*. Vol. 1839. 1. AIP Publishing LLC. 2017, p. 020202.
  - [93] B. Lorbeer, A. Kosareva, B. Deva, D. Softić, P. Ruppel, and A. Küpper. “A-BIRCH: automatic threshold estimation for the BIRCH clustering algorithm”. In: *INNS conference on Big Data*. Springer. 2016, pp. 169–178.
  - [94] C. M. Pereira and R. F. De Mello. “TS-stream: clustering time series on data streams”. In: *Journal of Intelligent Information Systems* 42.3 (2014), pp. 531–566.
  - [95] Y. Mirsky, B. Shapira, L. Rokach, and Y. Elovici. “pcstream: A stream clustering algorithm for dynamically detecting and managing temporal contexts”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2015, pp. 119–133.
  - [96] D. O. Cardoso, F. M. França, and J. Gama. “Wcds: A two-phase weightless neural system for data stream clustering”. In: *New Generation Computing* 35.4 (2017), pp. 391–416.
  - [97] J. de Andrade Silva, E. R. Hruschka, and J. Gama. “An evolutionary algorithm for clustering data streams with a variable number of clusters”. In: *Expert Systems with Applications* 67 (2017), pp. 228–238.
  - [98] J. P. Barddal, H. M. Gomes, and F. Enembreck. “SNCStream: A social network-based data stream clustering algorithm”. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 2015, pp. 935–940.
  - [99] J. P. Barddal, H. M. Gomes, F. Enembreck, and J.-P. Barthès. “SNC-Stream+: Extending a high quality true anytime data stream clustering algorithm”. In: *Information Systems* 62 (2016), pp. 60–73.
  - [100] L. Tu, P. Cui, and K. Tang. “A density grid-based clustering algorithm for uncertain data streams”. In: *2013 10th Web Information System and Application Conference*. IEEE. 2013, pp. 347–350.

- [101] H. Fichtenberger, M. Gillé, M. Schmidt, C. Schwiegelshohn, and C. Sohler. “BICO: BIRCH meets coresets for k-means clustering”. In: *European symposium on Algorithms*. Springer. 2013, pp. 481–492.
- [102] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler. “Streamkm++ a clustering algorithm for data streams”. In: *Journal of Experimental Algorithmics (JEA)* 17 (2012), pp. 2–1.
- [103] X. H. Dang, V. Lee, W. K. Ng, A. Ciptadi, and K. L. Ong. “An EM-based algorithm for clustering data streams in sliding windows”. In: *International conference on database systems for advanced applications*. Springer. 2009, pp. 230–235.
- [104] P. S. Bradley and U. M. Fayyad. “Refining initial points for k-means clustering.” In: *ICML*. Vol. 98. Citeseer. 1998, pp. 91–99.
- [105] D. Puschmann, P. Barnaghi, and R. Tafazolli. “Adaptive clustering for dynamic IoT data streams”. In: *IEEE Internet of Things Journal* 4.1 (2016), pp. 64–74.
- [106] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer. “An effective evaluation measure for clustering on evolving data streams”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011, pp. 868–876.
- [107] M. Moshtaghi, J. C. Bezdek, S. M. Erfani, C. Leckie, and J. Bailey. “Online cluster validity indices for performance monitoring of streaming data clustering”. In: *International Journal of Intelligent Systems* 34.4 (2019), pp. 541–563.
- [108] O. A. Ibrahim, J. M. Keller, and J. C. Bezdek. “Evaluating evolving structure in streaming data with modified Dunn’s indices”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2019).
- [109] L. E. B. Da Silva, N. M. Melton, and D. C. Wunsch. “Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study”. In: *IEEE Access* 8 (2020), pp. 22025–22047.

- 
- [110] H.-P. Kriegel, P. Kröger, I. Ntoutsi, and A. Zimek. “Density based subspace clustering over dynamic data”. In: *International conference on scientific and statistical database management*. Springer. 2011, pp. 387–404.
  - [111] H. Wang, Y. Yu, Q. Wang, and Y. Wan. “A density-based clustering structure mining algorithm for data streams”. In: *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 2012, pp. 69–76.
  - [112] G. Sheikholeslami, S. Chatterjee, and A. Zhang. “WaveCluster: a wavelet-based clustering approach for spatial data in very large databases”. In: *The VLDB Journal* 8.3 (2000), pp. 289–304.
  - [113] N. Ismael, M. Alzaalan, and W. M. Ashour. “Improved multi threshold birch clustering algorithm”. In: *International Journal of Artificial Intelligence and Application for Smart Devices IJAIASD* 2.1 (2014).
  - [114] P. P. Rodrigues, J. Gama, and J. P. Pedroso. “ODAC: Hierarchical clustering of time series data streams”. In: *Proceedings of the 2006 SIAM international conference on data mining*. SIAM. 2006, pp. 499–503.
  - [115] P. P. Rodrigues, J. Gama, and J. Pedroso. “Hierarchical clustering of time-series data streams”. In: *IEEE transactions on knowledge and data engineering* 20.5 (2008), pp. 615–627.
  - [116] S. Marsland, J. Shapiro, and U. Nehmzow. “A self-organising network that grows when required”. In: *Neural networks* 15.8-9 (2002), pp. 1041–1058.
  - [117] M. Bolaños, J. Forrest, and M. Hahsler. “Clustering Large Datasets Using Data Stream Clustering Techniques”. In: *Data Analysis, Machine Learning and Knowledge Discovery*. Ed. by M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning. Cham: Springer International Publishing, 2014, pp. 135–143. ISBN: 978-3-319-01595-8.
  - [118] Y. Prudent and A. Ennaji. “An incremental growing neural gas learns topologies”. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. Vol. 2. IEEE. 2005, pp. 1211–1216.

- [119] M.-R. Bouguelia, Y. Belaid, and A. Belaid. “An adaptive incremental clustering method based on the growing neural gas algorithm”. In: *2nd International Conference on Pattern Recognition Applications and Methods-ICPRAM 2013*. SciTePress. 2013, pp. 42–49.
- [120] C. Yin, L. Xia, S. Zhang, R. Sun, and J. Wang. “Improved clustering algorithm based on high-speed network data stream”. In: *Soft Computing* 22.13 (2018), pp. 4185–4195.
- [121] J. Gama, P. P. Rodrigues, and L. Lopes. “Clustering distributed sensor data streams using local processing and reduced communication”. In: *Intelligent Data Analysis* 15.1 (2011), pp. 3–28.
- [122] J. Chen, P. Chen, et al. “A Sketch-based clustering algorithm for uncertain data streams”. In: *Journal of Networks* 8.7 (2013), p. 1536.
- [123] C. C. Aggarwal. “On high dimensional projected clustering of uncertain data streams”. In: *2009 IEEE 25th International Conference on Data Engineering*. IEEE. 2009, pp. 1152–1154.
- [124] E. Lughofer. “A dynamic split-and-merge approach for evolving cluster models”. In: *Evolving systems* 3.3 (2012), pp. 135–151.
- [125] S. Guha, N. Mishra, R. Motwani, and L. o’Callaghan. “Clustering data streams”. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE. 2000, pp. 359–366.
- [126] D. K. Tasoulis, G. Ross, and N. M. Adams. “Visualising the cluster structure of data streams”. In: *International Symposium on Intelligent Data Analysis*. Springer. 2007, pp. 81–92.
- [127] M. Tavallaei, E. Bagheri, W. Lu, and A. Ghorbani. “A detailed analysis of the KDD CUP 99 data set”. In: *IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA* 2 (July 2009). DOI: 10.1109/CISDA.2009.5356528.
- [128] U. von Luxburg, R. C. Williamson, and I. Guyon. “Clustering: Science or Art?” In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR, Feb. 2012, pp. 65–79. URL: <https://proceedings.mlr.press/v27/luxburg12a.html>.

- [129] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. “GeoLife2. 0: a location-based social networking service”. In: *2009 tenth international conference on mobile data management: systems, services and middleware*. IEEE. 2009, pp. 357–358.
- [130] M. Garofalakis, J. Gehrke, and R. Rastogi. *Data stream management: processing high-speed data streams*. Springer, 2016.
- [131] B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams. “An Agent-Based Approach to Care in Independent Living”. In: *Ambient Intelligence*. Ed. by B. de Ruyter, R. Wichert, D. V. Keyson, P. Markopoulos, N. Streitz, M. Divitini, N. Georganas, and A. Mana Gomez. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–186. ISBN: 978-3-642-16917-5.
- [132] P. Domingos and G. Hulten. “A general method for scaling up machine learning algorithms and its application to clustering”. In: *In proceedings of the eighteenth international conference on machine learning*. Citeseer. 2001.
- [133] C.-H. Cheng, A. W. Fu, and Y. Zhang. “Entropy-based subspace clustering for mining numerical data”. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999, pp. 84–93.
- [134] T. Smith and D. Alahakoon. “Growing Self-Organizing Map for Online Continuous Clustering”. In: *Foundations of Computational Intelligence Volume 4: Bio-Inspired Data Mining*. Ed. by A. Abraham, A.-E. Hassanien, and A. P. d. L. F. de Carvalho. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 49–83. ISBN: 978-3-642-01088-0. DOI: [10.1007/978-3-642-01088-0\\_3](https://doi.org/10.1007/978-3-642-01088-0_3).

Algorithm	Window	Evaluation	CVIs	Dataset
BIRCH [30]	landmark	Comparison against K-MEANS and CLARANS. Entire datasets.	Run time, average pairwise distance within cluster, number of dataset scans, visualization	Self-made synthetic
CluStream [29]	pyramidal	Comparison against STREAM. Window evaluation.		
ClusTree [61]	damped	Comparison against DenStream and ClusTree. Entire datasets.	Weighted average purity, Median radius, Points Per second	KDD-CUP'99, CUP'98, synthetic
D-Stream [27]	damped	Comparison against CluStream. Window evaluation.	SSQ and run time	Covertype, Self-made synthetic
DenStream [28]	damped	Comparison against CluStream. Window evaluation.	Average purity, run time, memory usage	KDD-CUP'99, synthetic
DGClust [121]	landmark	Comparison against Continuous K-means. Entire datasets.	Loss against true cluster center	KDD-CUP'99, CUP'98, PDMC Sensor, synthetic
landmark	Comparison against KMeans and DI-ANA. Entire dataset.	MHT, DVI, CPCC, run time, memory usage	PDMC Sensor, Electrical Demand, self-made synthetic, Scalable k-means [31, 104]	Self-made synthetic <sup>a</sup> , ODAC [114, 115] landmark
Comparison against k-means and on-line k-means. Entire datasets.	Distance to ground truth centroids, loglikelihood, weighted entropy	Self-made synthetic, REV Digits Recognition, Reuters Information Retrieval dataset,		

<sup>a</sup> Self-made via generator from [132]

Algorithm	Window	Evaluation	CVIs	Dataset
Single pass k-means [32]	landmark	Comparison against Scalable k-means and k-means. Entire datasets.	SSQ, run time.	KDD-CUP'98, Self-made synthetic
Stream [125]	landmark	None	None	None
Stream LSearch [33]	landmark	Comparison against k-means and BIRCH. Window evaluation.	SSQ, run time, memory usage	KDD-CUP'99, Selfmade synthetic
StreamKM++ [102]	landmark	Comparison against BIRCH, STREAM LSearch, k-means, k-means++. Entire datasets.	Run time, SSQ	Spambase, KDD-CUP'99 (subset), Covertype, Tower, Census1990, BigCross <sup>a</sup> , Self-made synthetic
sliding	Comparison against CluStream. Window	SSQ, average purity, run time, memory usage	SWClustering [77] damped	KDD-CUP'99, KDD-CUP'98, Self-made synthetic HPStream [54]
Comparison against CluStream. Window evaluation.	Average purity, points processed per second	KDD-CUP'99, Covertype, and Self-made synthetic.		
E-Stream [47]	damped	Comparison against HPStream. Assuming window evaluation, not defined.	F-Measure, average purity, run time	Self-made synthetic

<sup>a</sup> Cartesian product of Tower and Covertype

Algorithm	Window	Evaluation	CVIs	Dataset
REPSTREAM [65]	damped	Comparison against (incremental)DBSCAN, HPSstream, CluStream, and DenStream. Window evaluation.	Average purity, entropy, and run time.	KDD-CUP'99, Covertype, and synthetic data.
OPTICS-Stream [126]	damped	Self analysis.	Reachability distance and ordering <sup>a</sup> .	Covertype, Self-made synthetic.
incPreDecon [110]	landmark	Self analysis.	run time	Selfmade synthetic, Bavarian Newborn screening
damped	Comparison against D-Stream.	Average purity, memory usage, update time.	KDD-CUP'99, Self-made synthetic	MR-Stream [46]
CellTree [90]	landmark	Comparison against DBSCAN. Window evaluation.	Accuracy, memory usage	Self-made synthetic <sup>b</sup>
Cell*Tree [44]	damped	Comparison to CellTree, STING, LSEARCH.	Accuracy, memory usage, update time.	Self-made synthetic <sup>c</sup>
SWEM [103]	sliding & damped <sup>d</sup>	Comparison against CluStream. Window evaluation.	SSQ, loglikelihood	Self-made synthetic
GWR [116]	Landmark	Comparisons against GNG and RCE. Entire datasets.	-	Self-made synthetic

<sup>a</sup> Based from OPTICS and is a criteria to create and manage clusters.

<sup>b</sup> Using generator from [133]

<sup>c</sup> Using generator from [133]

<sup>d</sup> A sliding window approach, but when elements fall out of the window a decaying factor is applied rather than instant removal.

Algorithm	Window	Evaluation	CV1s	Dataset
IGNG [118]	Landmark	Comparison against GNG and LVQ. Entire datasets.	Recognition rate	NIST, Iris, Self-made synthetic AIGNG [119]
	Comparison against Kmeans, SOINN, I2GNG.	Recognition rate and V-measure. against Kmeans, SOINN, I2GNG. En- tire datasets.	Self-made synthetic, Pendigit, Optdigit, Mnist, and documentary datasets. G-Stream [40, 87]	
	Average purity, NMI, Rand Index, run time.	DS1 <sup>a</sup> ,letter4 <sup>b</sup> , KDD-CUP'99, Covertype HUE-Stream [63]	damped	Comparison against UMicro. Window evaluation.
	KDD- CUP'99			
StrAP [88]	sliding	Comparison against DenStream. Window validation.	Error-rate, run time, average purity.	KDD-CUP'99, EGEE
SOStream [64]	damped	Comparison against MR-Stream and D- Stream. Window validation.	Average purity, memory usage, run time.	KDD-CUP'99, Self-made synthetic
SVStream [53]	sliding	Comparison against StreamKM, Clus- stream, DenStream, RepStream and StrAP. Window evaluation and entire datasets.	Rand Index, run time, memory usage.	Self-made synthetic, KDD-CUP'99, Cover- type

<sup>a</sup> Generated by <http://impca.curtin.edu.au/local/software/synthetic-data-sets.tar.bz2>

<sup>b</sup> Generated by <https://github.com/fel-dob/Token-Cluster-Generator>

Algorithm	Window	Evaluation	CVIs	Dataset
PreDeCon-Stream [57]	damped	Comparison against HPStream and Den-Stream. Window evaluation.	Average purity, run time	SynStream3D, Self-made synthetic, KDD-CUP'99, PDMC
UMicro [73]	pyramidal & damped	Comparison against CluStream	Average purity, points per second	KDD-CUP'99, Cover-type, Self-made synthetic
GCHDS [74]	damped	Comparison against HPStream. Window evaluation.	Average purity, points per second	KDD-CUP'99
GSCDS [75]	damped	Comparison against GCHDS. Window evaluation.	Average purity and run time	KDD-CUP'99, Cover-type, Self-made synthetic
rDenStream [59]	damped	Comparison against DenStream. Window validation.	Average purity, detection rate, memory usage	KDD-CUP'99, Cover-type, Self-made synthetic
DSCLU [76]	damped	Comparison against D-Stream and Den-Stream. Window evaluation.	Average purity, run time	KDD-CUP'99, Self-made synthetic
OPCluStream [111]	landmark	Comparison against OPTICS and OLD-Stream. Entire datasets.	run time, visualization	SEQUOIA (subset), Geolife, Self-made synthetic
CluDiStream [36]	landmark & sliding	Comparison against Scalable EM (SEM)	Average loglikelihood, run time, memory usage	NFD dataset, Self-made synthetic
BICO [101]	landmark	Comparison against StreamKM++, StreamLS, Birch, and MacQueen's k-means. Entire datasets.	SSQ, run time	Tower, Covertype, Census, BigCross, CalTech128

Algorithm	Window	Evaluation	CVIs	Dataset
DBSTREAM [41]	damped	Comparison against Stream, and CluStream. Prequential evaluation.	ARI, average purity, SSQ, memory usage, run time.	Self-made synthetic <sup>a</sup> , Sensor, KDD-CUP'99
HCluStream [35]	Pyramidal	Comparison against CluStream. Window evaluation.	Average purity	KDD-CUP'99
Olindda [34]	landmark	Comparison against other novelty detection algorithms	Error rates	Bloomed, Breast, Ionosphere, Mushroom, Leukemia, Lymphoma, Balance, and Iris. KDD-CUP'99
HDenStream [60]	damped	Comparison against DenStream. Window evaluation.	Average purity	
LiarTree [62]	damped	Comparison against ClusTree, DenStream, and CluStream. Window evaluation and entire datasets.	F1 measure, points per second, cluster radius, SSQ	Self-made synthetic, PDMC, Covertype
LeaDen-Stream [39]	damped	Comparison against DenStream, MR-Stream, and D-Stream. Unknown evaluation method.	Average purity, run time	KDD-CUP'99, Self-made synthetic
Improved BIRCH [113]	landmark	Comparison against BIRCH. Entire datasets.	Size of CF-Tree	Statlog, Abalone, Self-made synthetic

<sup>a</sup> Using generators from stream and MOA.

Algorithm	Window	Evaluation	CVIs	Dataset
A-BIRCH [93] evoStream [42]	landmark damped	Self analysis Comparison against DBSTREAM, D- Stream, DenStream, Clustream, Clus- Tree, STREAM, STREAMKMP++, and BICO. Prequential evaluation. Comparison against BIRCH and CURE. Entire datasets.	Accuracy, run time SSQ, run time	Self-made synthetic Powersupply <sup>a</sup> , data stream <sup>b</sup> , KDD-CUP'99, Covertype
Fractal Clustering [91]	landmark	Compared against EMicro and UMicro. Window evaluation.	Accuracy, run time	Selfmade synthetic
DCUSStream [48]	damped	Comparison against DenStream and D- Stream. Window evaluation.	Run time	Covertype, synthetic
HDCStream [49]	damped	Comparison against CFR. Win- dow evaluation.	Average purity, NMI, run time, memory usage.	KDD-CUP'99, Self-made synthetic
ICFR [51]	damped	Comparison against Weighted CFR. Win- dow evaluation.	Precision, recall, run time.	Weaver data in Japan.
WStream [52]	damped	Comparison against HPStream and Den- Stream. Window evaluation.	Average purity, run time.	Covertype, KDD-CUP'99
SiblingTree [55]	damped	Comparison against CLIQUE, lsearch, and hybrid-partition. Progression and entire datasets.	Accuracy, memory usage, processing time, coverage rate.	Self-made synthetic <sup>c</sup>
HDDStream [56]	damped	Comparison against HPStream. Window validation.	Average purity, memory usage	KDD-CUP'99, Cover- type

<sup>a</sup> [http://www.cs.ucr.edu/~eamonn/time\\_series\\_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/)

<sup>b</sup> <http://db.csail.mit.edu/labdata/labdata.html>

<sup>c</sup> Created using generator from ENCLUS [133].

Algorithm	Window	Evaluation	CVIs	Dataset
DUC-Stream [43]	damped	Comparison against STREAM and CluStream. Entire datasets.	Run time, memory usage, SSQ	KDD-CUP'99
DDStream [45]	damped	Comparison against CluStream. Evaluation method not specified, assuming window validation.	Accuracy, run time.	KDD-CUP'99
SDStream [37]	sliding	Comparison against CluStream. Evaluation method not specified, assuming window validation.	Average purity	KDD-CUP'99
FlockStream [38]	damped	Comparison against DenStream. Window evaluation.	Average purity, NMI	KDD-CUP'99, Self-made synthetic <sup>a</sup> .
C-DenStream [58]	damped	Comparison against DenStream. Window evaluation.	Rand Index	KDD-CUP'98, KDD-CUP'99, Self-made synthetic
DENGRISS-Stream [66]	sliding	None	None	None
MuDi-Stream [50]	damped	Comparison against DenStream. Window evaluation.	Weighted average purity, NMI, Rand Index, ARI, Folkes and Mallow index, Jaccard Index, F-measure, run time, and memory usage.	KDD-CUP'99, Landsat satellite, Self-made synthetic <sup>b</sup>
TS-Stream [94]	damped	Comparison against ODAC.	ARI	Stock market, Self-made synthetic
pcStream [95]	sliding <sup>c</sup>	Comparison against CluStream, DenStream, ClusTree, and tNN. Entire datasets.	ARI	KDD-CUP'99, Human Activity Recognition, HearO

<sup>a</sup> Based on datasets created by Cao et al. [28]

<sup>b</sup> Based on datasets created by Cao et al. [28].

<sup>c</sup> They denote their temporal filtering *drift buffer*, which is reminiscent of a sliding window.

Algorithm	Window	Evaluation	CVIs	Dataset
StreamXM [89]	landmark	Comparison against StreamKM. Entire datasets.	SSQ, weighted average purity	KDD-CUP <sup>99</sup> , Self-made synthetic <sup>a</sup>
SNCStream [98]	damped	Comparison against CluStream, ClusTree, DenStream. Entire datasets.	CMM, run time, RAM-hours	Self-made synthetic <sup>b</sup> , Airlines, Electricity, Covertype, KDD-CUP <sup>98</sup> , KDD-CUP <sup>99</sup>
SNCStream <sup>+</sup> [99]	damped	Comparison against CluStream, ClusTree, DenStream, HAStream and SNCStream. Entire datasets.	CMM, run time, RAM-hours, #of distance computations	Airlines, Electricity, Covertype, KDD-CUP <sup>99</sup> , BPoM, Self-made synthetic <sup>c</sup>
EDDS [85]	damped	Comparison against DenStream. Entire datasets.	Average purity, entropy, SSE, memory usage, run time, V-measure, ARI, NMI, #of clusters, entropy	Self-made synthetic <sup>c</sup>
WCDS [96]	sliding	Comparison against Agglomerative, K-Means, HD-BSCAN, ClusTree, CluStream, DenStream, ClusTree, CNDenStream, SNCStream <sup>+</sup> . Window evaluation.	KDD-CUP <sup>99</sup> , NID, Covertype, Gas sensors [130]	KDD-CUP <sup>99</sup> , NID, Covertype, Gas sensors [130]
WaveCluster [112]	landmark	Comparison against BIRCH and CLARANS. Entire datasets.	run time, visualization	Self-made synthetic <sup>d</sup>
Adaptive streaming k-means [105]	sliding	Comparison against CluStream and DenStream	Silhouette, number of clusters	Self-made synthetic <sup>e</sup> , traffic data from Aathaus <sup>f</sup>

<sup>a</sup> Using BIRCHCluster as a generator.<sup>b</sup> Using the RBF generator from MOA.<sup>c</sup> Using the RBF generator from MOA.<sup>d</sup> Some datasets were made by themselves, some by the authors of BIRCH [30].<sup>e</sup> <https://github.com/UniSurreyIoT/random-LHS-generator-drift><sup>f</sup> <http://www.oda.dk/dataset/realtids-trafikdata>

Algorithm	Window	Evaluation	CVIs	Dataset
FEAC-Stream [97]	damped	Comparison against StreamKM++. Window evaluation.	ARI, Simplified Silhouette, update time	Self-made synthetic, KDD-CUP'99, Cover-type, Localization data for Person Activity [131]
CEDAS [78]	damped <sup>a</sup>	Comparison against CluStream, DenStream, and MR-Stream. Window evaluation. Comparison against DenStream. Entire dataset.	Update time, average purity, accuracy, number of clusters, memory usage	Self-made synthetic, KDD-CUP'99
Improved stream clustering algorithm [120]	damped	Comparison against ECM. Entire datasets.	Accuracy, detection accuracy, false alarm rate, false negative rate,	KDD-CUP'99
DBIECM [79]	landmark	Comparison against HASTREAM, MRSream, and DenStream. Window evaluation.	Accuracy, average purity, Davies-Bouldin Index, SSE	Iris, Wine, Seeds, Glass, Breast Cancer.
I-HASTREAM [80]	damped	Comparison against CluStream and HP-Stream. Window evaluation, although not specified.	Average purity, CMM, run time	KDD-CUP'99, PDMC
ExCC [81]	sliding	Comparison against CluStream. Window evaluation.	Update time, average purity	KDD-CUP'99, Cover-type
GDStream [92]	damped	Comparison against CluStream. Window evaluation.	Accuracy, run time	KDD-CUP'99

<sup>a</sup> Linearly damped window

Algorithm	Window	Evaluation	CVIs	Dataset
Clu-US [100]	damped	Comparison against UMicro. Entire datasets.	SSQ, run time	Self-made synthetic, KDD-CUP'99
SWCUStream [82]	sliding	Comparison against CluStream. Window evaluation.	Average purity, run time	Covertype, CUP'99
UESSstream [122]	sliding	Comparison against UK-means. Entire datasets.	SRAND	Gensus 1990, Tower, Covertype
cluUS [86]	sliding	Comparison against CluStream-SW and UMicro-SW. Window evaluation.	Average purity, update time, memory usage	KDD-CUP'99, Self-made
HU-Clustering [83]	sliding	Comparison against UMicro. Window evaluation.	Average purity, run time	synthetic, IP
SWHU-Stream [84]	sliding	Comparison against UMicro. Self analysis. Entire datasets.	Run time, average purity	Covertype, CUP'99
UPStream [123]	damped	No access	RMS	KDD-CUP'99
GCP SOM [134]	No access	No access	No access	No access

Survey	Algorithms	Evaluation Method	CVIs	Dataset
Carnein et al. [23]	BIRCH, STREAM, CluStream, DenStream, DB-Stream, ClusTree, StreamKM++, BICO, and DB-STREAM.	Prequential evaluation	ARI for macro-clusters, average purity for micro-clusters. SSQ, weighted average purity, CMM # micro-clusters, macro-clusters, average purity, SSQ, cRand, silhouette, ARI, visualization of micro- and macro-clusters.	2-dim synthetic datasets, KDD-CUP99, Covertype, Sensor data stream <sup>a</sup> , Covertype, Poker Hand, Electricity, Adult data set. Self-made synthetic dataset, DSD_Benchmark <sup>#</sup> from the Stream Package in R 55, and Individual Household electric power consumption dataset <sup>b</sup> .
Mansalis et al. [24]	CluStream, DenStream and ClusTree	Window evaluation.		
Kokale et al. [13]	Sample k-means, sliding window k-means, D-Stream and DBSTREAM	Window evaluation.		

<sup>a</sup> <https://www.cse.fau.edu/~xqzhu/stream.html>

<sup>b</sup> <https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption>