Informatics

SDOstreamclust: Effective and Efficient Incremental Clustering of Streaming Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Mag.rer.nat. Simon Konzett
Matrikelnummer 0426743

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Senior Scientist Dr.techn. Félix Iglesias Vázquez
Mitwirkung: Univ.Prof. Dipl.-Ing. Dr.-Ing. Tanja Zseby

Wien, 27. November 2024

Simon Konzett   Félix Iglesias Vázquez

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Informatics

SDOstreamclust: Effective and Efficient Incremental Clustering of Streaming Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Mag.rer.nat. Simon Konzett
Registration Number 0426743

to the Faculty of Informatics

at the TU Wien

Advisor: Senior Scientist Dr.techn. Félix Iglesias Vázquez
Assistance: Univ.Prof. Dipl.-Ing. Dr.-Ing. Tanja Zseby

Vienna, November 27, 2024

Simon Konzett	Félix Iglesias Vázquez

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Mag.rer.nat. Simon Konzett

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 27. November 2024

Simon Konzett

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to express my gratitude to my supervisor, Félix Iglesias Vázquez, for his outstanding support, invaluable guidance, and insightful input throughout my work. His expertise and encouragement have been essential in shaping this project to meet both our expectations. Additionally, I would like to thank my co-supervisor, Tanja Zseby, for her constant support and for the valuable lessons she provided, which ultimately led me to this topic and field of study.

In a separate note, I would like to express my thanks to my partner, family, and friends for their ongoing support and understanding throughout this journey. To my partner, thank you for your continued encouragement and patience. To my family, I appreciate your belief in me and your emotional support. To my friends, thank you for the good times and for helping me stay balanced. I am truly grateful to have you all in my life.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

In einer Welt, die zunehmend von Daten geprägt ist, erzeugen Anwendungen wie die Netzwerkverkehrsanalyse, Finanzanalytik und die Überwachung von IoT-Geräten kontinuierlich sich entwickelnde Daten, die eine Echtzeitverarbeitung erfordern, um verwertbare Erkenntnisse zu gewinnen. Um die zugrunde liegenden Strukturen dieser Daten in dynamischen Umgebungen zu verstehen, müssen unüberwachte Methoden entscheidende Herausforderungen des Stream-Clustering bewältigen: den Umgang mit nicht-stationärem Verhalten, die Anpassung an sich ändernde Verteilungen und die Erkennung neu auftretender Klassen. Diese Phänomene, zusammenfassend als Konzeptdrift bezeichnet, stellen erhebliche Herausforderungen für traditionelle Methoden dar. Mit SDOstreamclust stellen wir einen Stream-Clustering-Algorithmus vor, der auf den Stärken von Sparse Data Observers aufbaut und dabei hohe Genauigkeit, geringen Rechenaufwand und Anpassungsfähigkeit an sich ständig verändernde Daten kombiniert. SDOstreamclust wurde speziell für dynamische Umgebungen und Echtzeit-Datenstreams entwickelt und zeichnet sich durch Skalierbarkeit, einfache Interpretierbarkeit, Effizienz und Robustheit bei minimaler Parametrisierung aus. Umfangreiche Experimente mit verschiedenen realen und synthetischen Datensätzen, ergänzt durch Analysen zur Parametersensitivität, belegen die hervorragende Leistung und Zuverlässigkeit im Vergleich zu etablierten Methoden. Besonders hervorzuheben ist, dass SDOstreamclust keine aufwendige Feinabstimmung erfordert und Konzeptdrift – eine häufig übersehene, aber zentrale Herausforderung realer Anwendungen, die oft zu einer schnellen Verschlechterung von Modellen führt – erfolgreich adressiert. Damit etabliert sich SDOstreamclust als eine leistungsstarke und verlässliche Lösung im Bereich der unüberwachten Analyse von Datenstreams.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

In a world increasingly driven by data, applications such as network traffic analysis, financial analytics, and IoT device monitoring continuously generate evolving data, necessitating real-time processing to uncover actionable insights. To understand the underlying structures of this data in dynamic environments, unsupervised methods must address critical stream clustering challenges: managing non-stationary behavior, adapting to distribution shifts, and identifying emerging classes. These phenomena, collectively referred to as concept drift, pose significant challenges to traditional methodologies. We introduce SDOstreamclust, an algorithm for stream clustering built upon Sparse Data Observers, leveraging their strengths to deliver high accuracy, low computational cost, and adaptability to evolving data. Designed for dynamic environments and real-time data streams, SDOstreamclust ensures scalability, interpretability, efficiency, and robustness with minimal parameterization. Comprehensive experiments on diverse real and synthetic datasets, complemented by parameter sensitivity analyses, demonstrate its superior performance and reliability compared to state-of-the-art methods. Notably, SDOstreamclust performs exceptionally well without extensive fine-tuning, effectively addressing concept drift, a critical yet often overlooked challenge in real-world applications that leads to rapid model degradation. These qualities establish SDOstreamclust as a strong competitor in the field of unsupervised streaming data analysis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

In this chapter, we present the background of our research, discussing the motivation and necessity behind the study. We will outline our specific objectives and the methodology employed. Lastly, a brief overview of the thesis structure will be provided.

1.1 Background

In today's data-driven world, vast amounts of information are continuously generated in real time. Unlike traditional static datasets, streaming data requires immediate attention and processing. The shift from static analysis to dynamic, real-time data processing is crucial for modern applications, where timely insights are essential for maintaining competitiveness, ensuring security, and fostering innovation.

Streaming data analysis plays a pivotal role in applications where real-time processing and adaptation are essential. These applications include network security, where continuous monitoring of network traffic is necessary to detect and respond to potential threats promptly. Financial services also rely on data stream analysis for real-time fraud detection and high-frequency trading, where timely and accurate data processing can have significant financial implications. Additionally, in smart cities, data from various sensors and IoT devices must be analyzed in real-time to manage resources efficiently, monitor traffic, and ensure public safety. Moreover, in healthcare, continuous patient monitoring generates streaming data that needs to be analyzed instantly to provide timely medical interventions. Each of these applications highlights the importance of robust, adaptable, and efficient data stream analysis techniques.

Change, adaptation, and novelty are key to analysing streaming data, which often requires unsupervised or semi-supervised techniques to identify novel patterns in evolving datasets where labeled data is typically unavailable due to their dynamic nature. To effectively capture these changes, algorithms must possess several critical capabilities: (a)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

processing data incrementally to produce timely results, (b) swiftly adapting to shifts in the data, (c) scaling effectively to handle substantial data volumes, (d) maintaining a compact and fixed-size model, (e) detecting outliers, and (f) accommodating diverse data types [SFB$^+$13]. Stream clustering and stream anomaly detection emerge as the primary branches in streaming environments, focusing on organizing data and identifying anomalies, respectively.

In this thesis, we focus on stream clustering, specifically on organizing data into meaningful groups while adapting to shifts and changes in the data distribution over time. However, static clustering methods have significant limitations. Böhm et al. summarize these drawbacks, noting that clustering algorithms often focus on spherical or Gaussian clusters, are sensitive to outliers, and require user-defined thresholds and parameters [BFPP06]. These limitations persist in more recent studies, as highlighted in works such as [NN19] and [PGR18]. In streaming environments, the challenges are compounded, as incremental analysis and evolving data — particularly concept drift — introduce significant additional complexity. Nguyen et al. [NWN15] identify concept drift as a major challenge for stream clustering, emphasizing it as a critical area for improvement in the reviewed algorithms. This thesis aims to address these challenges, focusing on the development of more robust and adaptable stream clustering techniques that can effectively manage the dynamic nature of streaming data.

1.2 Motivation

The growth of internet-connected devices and the expansion of network infrastructures have significantly increased both the volume and speed of network traffic, making network security a critical area where vast amounts of data are produced in real time. Efficiently processing and analyzing these data streams is essential for maintaining robust security measures.

Network traffic flows, consisting of continuous streams of data packets transmitted across network nodes, are essential for identifying patterns, detecting anomalies, and preventing unauthorized access. However, the high throughput and variability of network traffic pose significant challenges for conventional data processing techniques. Intrusion Detection Systems (IDS) are critical in defending against cyber threats by analyzing network traffic to identify suspicious activities that may indicate security breaches. While traditional IDS often rely on signature-based detection methods limited by their dependence on known attack signatures, anomaly-based IDS leverage machine learning and clustering techniques to detect unusual patterns in data streams. This adaptive approach is vital for recognizing novel threats, underscoring the necessity for robust stream clustering algorithms that can effectively handle the dynamic nature of network data. Based on my experience, improving these techniques is crucial for enhancing the overall effectiveness of network security measures.

Trending topics in data analysis, such as deep learning and generative AI, predominantly excel in supervised settings. However, in many real-world applications, such as finance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

and IoT, labels are often unavailable, and the environment is rarely as static as we might hope. This oversight of the adaptive nature of data and applications can lead to models that suffer degradation over time as the data evolves. Therefore, it is essential to address these challenges, particularly through enhanced stream clustering techniques that can effectively identify patterns and navigate the complexities of a dynamic unsupervised environment.

1.3 Goals

In this section, we present the research questions guiding our investigation into our novel data stream clustering algorithm.

- **RQ1: Performance**

  How and to what extent can an incremental version of SDOclust improve the performance of consolidated stream clustering in terms of accuracy and execution times?

- **RQ2: Self-adjustment**

  To what extent can an incremental version of SDOclust effectively self-adjust to overcome the challenges of stream clustering, especially concept drift? Is it superior to state-of-the-art alternatives?

- **RQ3: Applicability**

  Can interpretability and parameter robustness be maintained in incremental clustering algorithms while ensuring a lightweight profile? To what extent can the models and parameters of SDOstreamclust be considered interpretable and robust?

1.4 Methodology

To achieve those goals the following steps were done:

- We built our algorithm upon the foundations of SDOstream [HIZ20] and SDOclust [IZHZ23], conducting a comprehensive study of their features, underlying algorithms, relevant literature, and existing Python implementations.

- We designed SDOstreamclust to achieve high accuracy and low computational cost, which are essential for dynamic environments. Our design goals include scalability to handle unlimited data streams within resource and time constraints; *adaptability* to evolving data patterns; *efficiency* for lightweight, real-time processing; *interpretability* to provide clear insights into clusters; an almost *parameter-free* approach to minimize the need for manual adjustments; and *robustness* to ensure stability over time, avoiding unexpected performance drifts and high sensitivity to parameter changes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- We established a test environment by selecting competitor algorithms, datasets with concept drift, and appropriate evaluation metrics to ensure workflow compatibility. Our approach focused on various datasets, particularly those involving evolving data, to emphasize the importance of robustness in parameterization. A data stream algorithm must maintain performance without manual intervention amid changing data, making this a key aspect of our evaluation.

- We conducted comprehensive evaluation experiments, including comparisons with competitor algorithms and detailed parameter analyses. We collected and analyzed the results to gain insights, propose refinements, and outline potential improvements for future work.

- We documented the process in this thesis, and a conference paper is under review [1], making SDOstreamclust available via `pip` [2] or directly from the repository [3], with evaluation experiments reproducible through Docker [4]. Detailed setup instructions are provided in a DOI-citable repository [IV24].

1.5 Structure

The thesis is structured as follows: Chapter 2 provides relevant background information, covering topics such as data stream processing, state-of-the-art methods and approaches in stream clustering, as well as challenges and strategies for handling evolving, changing, or anomalous data in streaming scenarios. Chapter 3 describes our novel method, giving algorithmic and implementation details in Section 3.1, and elaborating on the evaluation setup (including the datasets, comparison algorithms metrics, and experiments) in Section 3.2. The results are presented and discussed in Chapter 4, and the thesis concludes with a summary of findings and future research directions in Chapter 5.

[1]Félix Iglesias Vázquez, Simon Konzett, Tanja Zseby, and Albert Bifet (2024). Stream Clustering Robust to Concept Drift. Under review at the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2024*

[2]`https://pypi.org/project/pysdoclust-stream/`

[3]`https://github.com/CN-TU/pysdoclust-stream/`

[4]`https://hub.docker.com/r/fiv5/sdostreamclust`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Background Knowledge

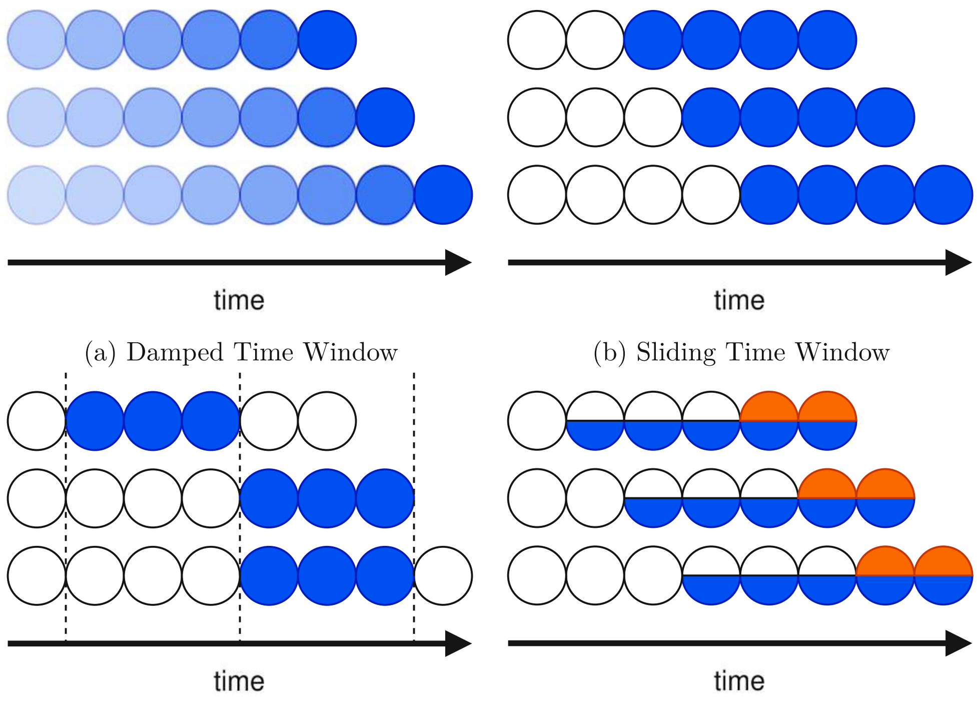
This chapter provides an essential foundation on data stream processing and clustering, focusing on its significance and the primary methodologies employed to tackle real-time data analysis. It introduces key approaches such as partitioning, hierarchical, density-based, and grid-based methods, and addresses the unique challenges posed by the dynamic and continuous nature of data streams.

2.1 Data stream processing

One of the key challenges in stream clustering is maintaining an up-to-date clustering model that accurately reflects the evolving data distribution. This requires algorithms to quickly assimilate new data points, detect and adapt to concept drifts, and discard obsolete information, all while operating within stringent memory and processing constraints. Additionally, ensuring the robustness and stability of clusters over time, despite the transient nature of streaming data, remains a significant hurdle.

Time window models are commonly used to manage to manage and process the continuous flow of data efficiently. They help in capturing the most relevant data points while addressing the challenges of unbounded data streams. Here, we explain the four most commonly used time window models [SFB+13], [NWN15], [CT19]:

Damped time window. In a damped time window model, recent data points are given more importance than older ones by applying a decay factor λ to the weight of each data point, which diminishes their significance over time. The weight of a data point can, for example, be defined as $w(\Delta t) = e^{-\lambda \Delta t}$, where Δt represents the time elapsed since the data point's arrival. As new data arrives, older data gradually loses its influence on the clustering process, allowing the model to quickly adapt to changes in the data stream while still considering past information to some extent. This approach is commonly used in popular algorithms such as DenStream [CEQZ06] and DBStream [HB16].



(a) Damped Time Window (b) Sliding Time Window

(c) Landmark Time Window (d) Tilted Sliding Time Window

Figure 2.1: Time Window Models: The figures illustrate different strategies for handling temporal data in streaming scenarios

Sliding time window. The sliding time window model maintains a continuously moving window of data points as new data arrives, operating with either a fixed-size or dynamically adjusted window. This model typically follows the first-in-first-out (FIFO) principle, where the oldest data points are discarded as the window shifts to accommodate new arrivals. By considering only on the data points within the current window, the clustering process reflects the most recent data, providing a dynamic snapshot of the evolving state of the data stream. This approach is used in algorithms such as SWClustering [ZCQJ08] and SDStream [RM09].

Landmark Time Window. In a landmark time window model, the data stream is divided into disjoint segments based on specific landmarks or significant events. All data points between two consecutive landmarks are considered for clustering, allowing analysis within these predefined intervals. This approach is particularly useful for algorithms that cannot evolve continuously and therefore require periodic restarts, or in situations where the data stream naturally exhibits distinct phases or cycles. It is employed in algorithms such as BIRCH [ZRL96] and Stream [GMMO00].

Tilted Time Window. The tilted time window model provides a multi-resolution view of the data stream by maintaining different time windows with varying granularity.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Individual windows can follow damped, sliding, or landmark models. It retains finer granularity for recent data and coarser granularity for older data, making it particularly effective for capturing both short-term trends and long-term patterns. For instance, it may store detailed information about the last few minutes, less detail for the past hours, and even less for the past days or weeks. This approach is used in algorithms such as CluStream [APHW03] and StreamKM++ [AMR+12].

These four models — damped time window, sliding time window, landmark time window, and tilted time window, as illustrated in Fig. 2.1 — represent strategies for handling continuous data streams. Each approach offers unique advantages in balancing the emphasis on recent data while maintaining the relevance of older information, effectively addressing the challenges posed by evolving data distributions in stream analysis.

2.2 Stream Clustering

Clustering is a fundamental task in data analysis, which involves grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups.

In recent years, the emergence of data streams — continuous flows of data generated in real-time — has necessitated the development of stream clustering methods. Stream clustering has emerged as a critical technique in the field of data mining, dedicated to the real-time analysis and classification of continuously flowing data. Unlike static datasets, data streams are characterized by their high velocity, potentially unbounded size, and dynamic nature, posing unique challenges for traditional clustering algorithms. To address these challenges, specialized stream clustering methods have been developed, each designed to efficiently process and organize data on-the-fly while managing resource constraints such as limited memory and computational power.

In this section, essential concepts of stream clustering are outlined, encompassing the most popular types, taxonomies, approaches, methodologies, and algorithms. The algorithms that are structural in this work (i.e., SDO, SDOclust, and SDOstream) are not explained here, but rather discussed in detail in Chapter 3, specifically in Section 3.1.1.

2.2.1 Core Principles and Taxonomies

Various stream clustering methods have been developed to handle different data types within streaming scenarios. However, optimizing these methods is constrained by factors such as the availability and order of data, as well as limitations in resources and time. Due to the large volume of data, storing all observations becomes impractical or infeasible. Typically, each observation should only be evaluated once before being discarded, necessitating the extraction of sufficient information from each observation. Similarly, the order of observations should not be disturbed and must be accommodated within the clustering process [CT19]. These constraints give rise to various considerations regarding the structure of stored data.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Nguyen et al. [NWN15] categorize data stream clustering methods into five types, similar to traditional data clustering: (a) *partitioning methods*, which divide the data into a predefined number of clusters, such as StreamKM++ [AMR+12]; (b) *hierarchical methods*, which create a tree-like structure of clusters by progressively merging smaller clusters (agglomerative) or splitting larger clusters (divisive) based on a measure of dissimilarity, exemplified by BIRCH [ZRL96]; (c) *density-based methods*, which form clusters based on regions of high data density, allowing for the discovery of clusters with arbitrary shapes, including DenStream [CEQZ06] and DBStream [HB16]; (d) *grid-based methods*, which partition the data space into cells and form clusters by grouping neighboring dense cells, as seen in D-Stream [CT07]; and (e) *model-based methods*, which assume a specific statistical model for the data and estimate its parameters to identify clusters, such as CluDistream [ZCY+06]. These categories are not mutually exclusive and can overlap. All clustering methods rely on a measure of distance or similarity between clusters. The four primary distance measures are: single-linkage, which is the minimum distance between any pair of points in the two clusters; complete-linkage, defined as the maximum distance between any pair of points; mean distance, which considers the distance between the centroids (e.g., arithmetic mean) of the clusters; and average distance, calculated as the mean of all pairwise distances between points in the two clusters. Due to the high computational cost associated with maximum distance and average distance measures, they are generally not suitable for data stream clustering applications.

Silva et al. [SFB+13] propose an alternative taxonomy for stream clustering algorithms. This taxonomy emphasizes different criteria or may organize them in a different manner compared to the one presented before. Data stream clustering algorithms typically consist of two main steps: the incremental data abstraction step and the subsequent clustering step, which often involves offline processing where a traditional static clustering algorithm is applied to the abstracted data. This taxonomy aspects include:

t1. the data structure used for statistical summary,

t2. the window model,

t3. the mechanism for outlier detection,

t4. the number of user-defined parameters,

t5. the (offline) clustering algorithm utilized,

t6. the shape of clusters formed,

t7. type of clustering problem addressed.

This taxonomy provides a structured framework for understanding and comparing various approaches to data stream clustering, allowing researchers to identify key characteristics of each method and assess their suitability for specific applications.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The first two characteristics (t1, t2) focus on the data abstraction of the stream, which are explored in more detail in Sections 2.1 and 2.2.2. The abstracted data is typically processed using established static clustering methods. The choice of clustering method (t5) influences the shape of the clusters that can be detected (t6). For example, k-means [Llo82], [M$^+$67], [Ste56], [BH$^+$65] tends to identify spherical clusters, while DBSCAN [EKS$^+$96] and hierarchical clustering can detect clusters of arbitrary shapes. Ideally, these methods should differentiate between true outliers and natural cluster evolution (t3), as further elaborated in Section 2.3.2.

Algorithms vary in the number of user-defined parameters (t4) required, such as the number of clusters, window size, and decay rate. Fewer parameters can simplify the algorithm, making it more user-friendly, but this may come at the cost of reduced flexibility and adaptability. However, ease of use is not solely determined by the number of parameters. The robustness of an algorithm concerning parameter choices and the intuitiveness of these parameters also significantly contribute to user-friendliness. While clustering problems can be either object-based or attribute-based (t7), this discussion focuses exclusively on object-based clustering.

2.2.2 Data structures

In stream analysis, data arrives continuously and in high volumes, making it impractical to store and process every single data point. Therefore, stream clustering algorithms need to summarize the data efficiently while preserving its essential statistical properties.

A statistically summarizing data structure allows stream clustering algorithms to capture the key statistical characteristics of the data, such as its distribution, centroids, or density, without retaining every individual data point. By summarizing the data statistically, algorithms can reduce memory usage and computational complexity while still providing meaningful insights into the underlying data distribution.

The main data structures are micro-clusters, grids, coresets, and prototype arrays. Each is explained as follows.

Micro-clusters. Micro-clusters provide a compact and efficient structure for summarizing data points in stream clustering. A key data structure, introduced by BIRCH [ZRL96], is the Clustering Feature (CF) vector, designed to handle unbounded and dynamic data streams in an incremental and memory-efficient manner. The CF vector consists of three components: N, the number of data objects; LS, the linear sum of the data objects; and SS, the sum of squared data objects. Both LS and SS are arrays with n dimensions. These components allow for incremental updates as new data points arrive, enabling the real-time computation of cluster properties such as mean, radius, and diameter.

Many algorithms employ these CF vectors, or similar structures, to form micro-clusters, which are compact representations of dense regions in the data. These micro-clusters are continuously updated to capture evolving patterns in the data stream and often serve as inputs to traditional clustering algorithms. This leads to two-phase clustering processes,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

where the first phase involves summarizing the data through micro-clusters, and the second phase applies a clustering algorithm to the summarized data. For example, BIRCH uses hierarchical clustering, CluStream [APHW03] and StreamKMeans [OMM$^+$02] use k-means, and DenStream [CEQZ06] employs DBSCAN, as noted in [HB16] and [CT18]. This approach allows for efficient, adaptive clustering of streaming data under real-time or memory-constrained conditions.

Grids. Another common data structure in stream clustering is to capture the density of observations within a grid structure. A grid partitions the data space along all dimensions into intervals, creating a number of grid-cells. By mapping data points to these cells, a density estimate can be maintained, allowing the algorithm to identify regions of high data concentration by grouping adjacent dense cells.

The main challenge for grid-based clustering algorithms lies in the construction of the grid-cells: specifically, determining how often cells should be partitioned and choosing the appropriate size for each cell. These decisions are crucial as they impact the granularity and accuracy of the clustering results.

A well-known grid-based algorithm is D-Stream [CT07], from which many variations have evolved. D-Stream is a density-based stream clustering algorithm that leverages grid-cells to manage data streams effectively.

Coreset. Coresets are representative subsets of a dataset that approximate the properties of the original dataset, primarily used to efficiently approximate the results of clustering and other computational tasks. A coreset reduces the size of the data while retaining enough information to ensure that computations performed on the subset yield results close to those obtained using the full dataset. The quality of a coreset depends on how well it represents the original data, and it is typically constructed through techniques such as sampling or optimization. By reducing the data size, coresets are highly scalable and enable more tractable computations, making them valuable for large-scale data processing, particularly in stream clustering.

Coreset trees, as employed by StreamKM++ [AMR$^+$12], combine concepts from feature vectors and prototype arrays within a binary tree structure to efficiently manage and process streaming data. In contrast, the method proposed in this work, SDOstreamclust, leverages a coreset-based approach but adopts a partitional strategy instead of the hierarchical tree structure used in StreamKM++.

Prototype Array. Prototype Arrays are specialized data structures used in clustering algorithms to represent clusters through a small set of representative points, such as centroids or medoids. Each prototype serves as a center in the feature space, capturing the key characteristics of a cluster. Unlike coresets, which provide a summary of the entire dataset, Prototype Arrays focus specifically on representing clusters with a limited number of points. This makes them particularly useful in scenarios where a small number

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

of prototypes can adequately describe the data distribution. As new data points arrive, the prototypes are updated iteratively, allowing the array to adapt to evolving clusters.

Prototype arrays are frequently used in algorithms for dynamic, large-scale clustering, as demonstrated in methods such as Stream [GMMO00] and LSearch [OMM+02].

In summary, four main types of data structures used in stream clustering are micro-clusters, grids, coresets, and prototype arrays. Each plays a crucial role in efficiently summarizing and managing data streams, enabling scalable and adaptive clustering solutions under real-time or resource-constrained conditions. These structures allow algorithms to balance memory usage, computational efficiency, and clustering accuracy when dealing with large and continuously evolving datasets.

2.3 Novelty, change and outlier detection

Silva et al. [SFB+13] point out that many data stream clustering algorithms overlook a crucial aspect of data stream mining: change detection. In addition to addressing evolving data streams and change detection, it is essential for clustering algorithms to effectively handle noise, or outliers, throughout the data stream [Bar02]. This capability is crucial as outliers can significantly impact clustering accuracy and the interpretation of evolving data patterns. Specifically, in this section we look into two key concepts: Concept Drift and Outlier Detection.

2.3.1 Concept Drift

It is well-known that the data generation processes for various stream applications are driven by nonstationary distributions. This phenomenon, known as concept drift, means that the underlying concept from which data is derived can shift periodically, each time after a minimum period of stability.

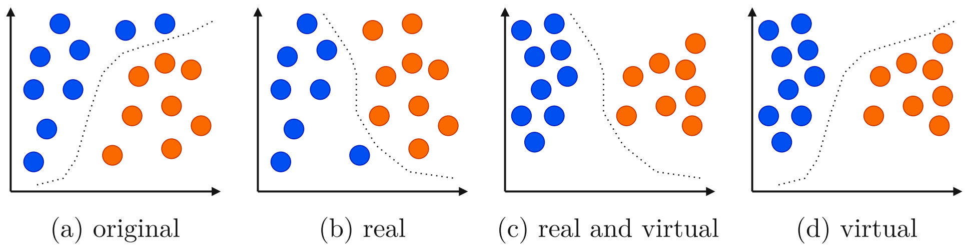

(a) original (b) real (c) real and virtual (d) virtual

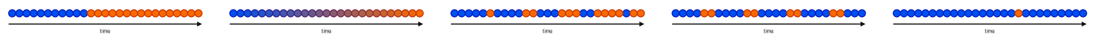
Figure 2.2: Concept Drift: Real and Virtual, based on [GŽB+14]

Gama et al. [GŽB+14] distinguish between two types of concept drift: real and virtual concept drift. Real concept drift happens when the classification boundaries change over time, affecting how data is classified. This type of drift can also occur alongside changes in the appearance or distribution of the data, making it harder to identify the underlying shifts in classification. In contrast, virtual concept drift refers only to changes

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

in the data's appearance or distribution while the classification itself remains the same. Figure 2.2 illustrates these concepts, demonstrating how real drift can occur without changes in data appearance, as seen in (b), while (c) shows real drift with changes in appearance. Understanding this distinction is important for recognizing how data streams evolve and for developing effective strategies to adapt to these changes in classification dynamics, particularly when real concept drift occurs together with virtual drift.

However, in practice, distinguishing between real and virtual concept drift can be challenging. The complexities of real-world data streams often make it hard to tell whether changes in data distribution indicate a fundamental shift in classification boundaries or simply reflect variations in data appearance. This confusion can complicate adaptation strategies, as misinterpreting the type of drift may lead to inappropriate responses to changes in the data.



(a) sudden (b) incremental (c) gradual (d) reoccuring (e) outlier

Figure 2.3: Data Evolution: Concept Drift and Anomalies

They also identify various patterns of concept drift, including (a) *sudden drift*, characterized by an abrupt change of states; (b) *incremental drift*, which involves a continuous and slow transition from one state to another; (c) *gradual drift*, where a new state gradually replaces an old state with increasing frequency until the old state disappears; and (d) *reocurring concepts*, in which states frequently appear and disappear. Additionally, another phenomenon that is not technically concept drift but may appear as such to an algorithm is the presence of (e) *noise* or *outliers*. These patterns are illustrated in Figure 2.3, which provides a visual representation of the different types of concept drift and the occurrence of outliers.

Moreno et al. [MTRAR$^+$12] adopt a theoretical perspective on concept drift to unify the diverse and often confusing terminology found in the literature. They define concept drift as changes in the distribution of data over time, which can lead to a deterioration in model performance. Consequently, models must be adapted to maintain their predictive accuracy in the face of such changes. They distinguish between two types of problems, based on the functional relationship between the features and the class:

- $X \to Y$: Features $x \in X$ causally determine the class label $y \in Y$, e.g., voting behavior is influenced by cultural background and economic situation. Joint distribution: $P(x, y) = P(y|x)P(x)$.
- $Y \to X$: The class label $y \in Y$ determines the features $x \in X$, e.g., recognizing a "Bird" implies wings and feathers. Diagnosing diseases also falls into this category. Joint distribution: $P(x, y) = P(x|y)P(y)$.

From this concept, the term *dataset shift* refers to a change in the joint distribution between training and test data. In unsupervised problems, such as stream clustering,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the test data corresponds to future incoming data, making it essential to adapt models rapidly to these changes. They distinguish four types of shifts: *covariate shift*, which occurs only in $X \to Y$ relationships where the distribution of the input data changes, but the relationship between features and labels remains unchanged; *prior probability shift*, the equivalent for $Y \to X$ problems, where the distribution of the class changes, but the relationship between the class and features remains fixed; and *concept drift*, where the relationship between features and class evolves, while the feature or class distribution stays constant for $X \to Y$ or $Y \to X$ problems. If both parts of the distribution change, the authors deem the problem extremely challenging and potentially unsolvable [MTRAR+12].

Relating these to earlier mentioned concepts of concept drift, the first two categories align with virtual drift, differentiated by the type of problem at hand. Real concept drift, as defined by [GŽB+14], covers not only changes in the feature-class relationship but also the more complex case where prior distributions shift as well.

In a data stream setup, these distinctions between dataset shifts are less clear, with significant ambiguity. There is no clear-cut answer for defining training and test data at any given moment—how far back should training data go, and how far into the future should test data extend? Additionally, real-world data often involves both $X \to Y$ and $Y \to X$ relationships to some extent. In unsupervised setups, prediction typically follows the $X \to Y$ process, where labels are determined based on features. However, a robust algorithm is expected to adapt only when necessary — by updating the model to reflect changes — implicitly addressing the $Y \to X$ relationship when relevant. Given this ambiguity, many scenarios may present as concept drift in unsupervised data stream analysis, making it crucial for methods to be adaptive and robust in handling various forms of drift.

2.3.2 Outlier Detection

The definition of an outlier is not straightforward. Hawkins [Haw80] provided a popular definition in colloquial speech: an outlier is *an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism*. Similarly, [JM06] describe outliers as objects that deviate from the general behavior of a data model, while [CBK09] define outliers as sparse, isolated data items that exhibit significantly different characteristics from normal data.

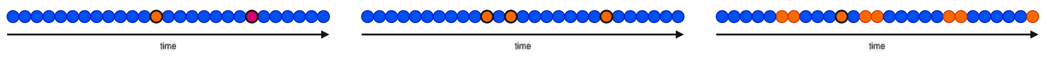
These definitions all rely on some notion of normality, which can be formalized in probability theory, as in early works by [Edg87] and many others since. In this context, normal behavior is modeled by a probability distribution, and outliers are identified as observations that fall within low-probability regions, defined by a sufficiently small threshold of the distribution, where the occurrence of normal data is unlikely.

Outlier detection is essential across various fields and significantly impacts daily life. It helps monitor login behavior and payment transactions to identify fraud and aids in medical diagnosis by detecting anomalies in patient data for early disease detection.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Additionally, it plays a crucial role in noise removal during data preprocessing, ensuring cleaner datasets for analysis. Outlier detection also supports scientific discovery by uncovering unusual patterns that may indicate new insights. In trending applications like natural language processing (NLP), computer vision, autonomous driving, and speech recognition, it enhances model performance and reliability. These examples showcase just a small sample of the diverse applications of outlier detection.

Outliers, anomalies, and novelties share similar characteristics and are often used interchangeably depending on the context. When distinguished, anomalies represent instances from a different source (e.g., fraud, hacker attacks), while outliers are typically noisy data points within the normal distribution (e.g., measurement errors) and may be removed during data cleaning. Novelties occur when the normal distribution shifts, either gradually or through new patterns. Although detection methods may be similar, their treatment varies by application. Here, we use "outlier" to refer to all these types, recognizing that specific contexts may favor one term over another.



(a) point (b) group (c) contextual

Figure 2.4: Illustrations of common outlier types in a streaming setup: (a) point outliers, which deviate individually, (b) group outliers, where multiple data points deviate together, and (c) contextual outliers, where data points become outliers based on their temporal context. Outlier points are marked with a bold edge.

Outliers can be characterized in various forms, as outlined by Ruff et al. [RKV+21], who distinguish between point, group, contextual, low-level sensory, and high-level semantic outliers. The latter two types are less relevant in this work, as they are more prevalent in applications like image recognition and natural language processing (NLP). The most straightforward type is individual instances, such as a sudden spike in sensor readings. In intrusion detection, hacker attacks often manifest as group outliers, where multiple data points deviate together. Relevant to this work on streaming data, contextual outliers frequently appear in time series, where certain instances may be common but become outliers when they occur at abnormal times. The point, group, and contextual outliers are visualized in Figure 2.4, within a streaming setup.

Due to the extensive applications and diverse types of outliers, numerous approaches to outlier detection have emerged across various fields. Ruff et al. [RKV+21] categorize these methods into shallow and modern deep learning approaches, further dividing them based on model types such as classification, probabilistic, reconstructive, or distance-based. A common characteristic in outlier analysis is the lack of labels, as outliers are often defined by their "non-normal" status relative to the dataset, which leads to a predominance of unsupervised methods. Given this variety, there is no one-size-fits-all solution; the choice of approach depends on the data type, context, and specific application requirements.

Clustering is closely related to outlier detection, as outliers are data points that do not

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

fit well into any known class. Many clustering algorithms leverage their capabilities to incorporate outlier detection, assigning either an outlier label or a score that quantifies the outlierness of analyzed points. In a streaming context, challenges related to data stream processing (Section 2.1) and evolving data (Section 2.3.1) are prevalent. Iglesias et al. [VHZZ23] examine popular outlier detection methods, particularly SDOstream [HIZ20], which serves as the foundation for the proposed clustering approach in this work, focusing on its performance in terms of *locality* (contextual outlierness), *relativness* (outliers relative to past data), *concept drift*, and its runtime and memory efficiency in streaming setups. The study shows that the SDO family of stream outlier detection methods is competitive with popular approaches, such as a sliding window version of the Local Outlier Factor (LOF) algorithm [BKNS00], Robust Random Cut Forests (RRCF) [GMRS16], RSHash [SA16], a hash-based technique, LODA [Pev16], and xStream [MLA18], which project data into a lower-dimensional space.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU Bibliothek WIEN Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Methodology and Experiments

In this chapter, we introduce our proposed methodology, detailing its theoretical underpinnings, practical implementation, and empirical evaluation. We explore the core aspects of our method, explaining its design principles and key parameters. Our approach uses advanced computational techniques to tackle challenges in data clustering and analysis, aiming to improve both accuracy and efficiency.

Our method is implemented in C++ with a Python wrapper for easy integration and use. This combination ensures high performance and flexibility, making it easy to compare with other methods. We evaluate its effectiveness through extensive testing on various datasets, demonstrating its robustness and versatility.

The following sections will describe our methodology, including the new algorithm, parameter choices, and implementation details. We will present experiments to test our method's performance and compare it with existing techniques, highlighting its strengths and areas for improvement. This chapter aims to provide useful insights into data clustering and its applications.

3.1 Design of SDOstreamclust

The here proposed method SDOstreamclust combines the incremental processing capabilities of SDOstream [HIZ20] with the robust clustering techniques used in SDOclust [IZHZ23] (see Section 3.1.1 for details on both). The resulting method leverages both the real-time adaptability of SDOstream and the structured clustering methodology of SDOclust, thereby offering a powerful approach for streaming data processing and clustering.

3.1.1 The Sparse Data Observers principle and precedent algorithms

Sparse Data Observers (SDO) [VZZ18] is an algorithm designed for outlier scoring and detection. It builds a low-density model of the data during the training phase, enabling quick analysis and autonomous decision-making. As an eager learner, SDO minimizes computational costs during application, making it well-suited for big data environments. Additionally, SDO is robust to parameter variations, ensuring reliable performance with minimal need for extensive prior knowledge.

In this section, we introduce SDO, which serves as the foundation for the novel algorithm proposed in this work. We will also cover its advancements in streaming and clustering applications, offering a detailed overview of the SDO family and its role in supporting the new method.

Table 3.1: Notation and Definitions

	General
$v_i \in \mathbb{R}^m$	Point with index i
$t_i \in \mathbb{R}$	Timestamp with index i
$s_i \in \mathbb{R}$	Outlier score of point with index i
$l_i \in \mathbb{R}$	Label of point with index i
	Observer Model
$k \in \mathbb{N}$	Number of observers
$x \in \mathbb{N}$	Number of neighbors to consider
$\rho \in (0,1]$	Fraction of observers to consider idle
$\omega \in \mathbb{R}^m$	An observer
Ω, Ω_a	Set of (active) observers
$\mathcal{N}, \mathcal{N}_a$	Set of x closest (active) observers
	Stream Processing
$f \in (0,1]$	Fading parameter
$P_\omega \in \mathbb{R}$	Observations by ω
$H_\omega \in \mathbb{R}$	Observer ω's age
$\overline{P}_\omega = \frac{P_\omega}{H_\omega}$	Average observations by ω
$L_\omega \in \mathbb{R}^{\lvert C \rvert}$	Cluster-observations by ω
	Connected Components Clustering
$\chi \in \mathbb{N}$	Local threshold for cutting-off graph edges
$\zeta \in [0,1]$	Weight factor regarding locality and globality in thresholds
$e \in \mathbb{N}$	Minimum number of observers that a cluster can have
$h_\omega, h'_\omega \in \mathbb{R}$	Local(-global weighted) cutoff threshold of the observer ω
$h \in \mathbb{R}$	Global density threshold
S	A cluster, i.e. a set of active observers
$\mathcal{C}, \mathcal{L}$	Set of (labeled) clusters
C, c	Set of unique cluster labels, a cluster label
$\mathbf{l}_\omega = \frac{L_\omega}{\sum_c L_\omega}$	Cluster membership vector of ω
	Outlier thresholding
h^o	relative outlier sensitivity parameter
$\mathbf{p}_i^o, p_i^o(\omega)$	outlier probability of data point v_i (w.r.t. observer ω)

Note: This table summarizes the key symbols and their definitions used throughout the paper.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 3.1 is provided to enhance readability and serve as a quick reference for readers.

SDO. The Sparse Data Observers (SDO) algorithm [VZZ18] is an effective method for outlier detection and scoring. It operates in two distinct phases: LEARNING and PREDICTION.

In the LEARNING phase, the algorithm builds a low-density model by selecting representative data points. During the PREDICTION phase, it computes the outlier score for new data points based on their distance from and density relative to the model. This approach enables efficient anomaly detection while ensuring computational efficiency, making it highly suitable for large-scale data applications.

The detailed steps of the SDO algorithm are as follows and illustrated in Fig. 3.1:

Learning

1. **Sample:** Randomly select k data points from the dataset. These points, referred to as *observers*, are used to build a low-density model that captures the data's statistical properties.
2. **Observe:** For each observer, count the number of points within its neighborhood, defined as the x nearest points. This count, known as *observations*, serves as a quality metric for the observers.
3. **Clean model:** Categorize observers into idle and active based on an observation threshold to exclude outliers and noise. Retain the top $1 - \rho$ fraction of observers as *active observers*, which are then used to construct the sparse, representative low-density model.

Prediction

1. The **outlier score** is calculated as the median distance between the data point in question and its x nearest active observers.

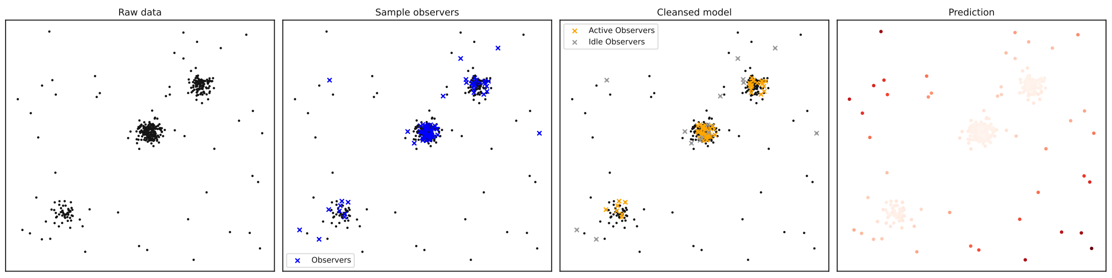


Figure 3.1: Illustration of the SDO algorithm, showcasing the four key steps involved. In PREDICTION, outlierness for points is colored according to their scores (redder indicates a higher outlier score).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The critical parameter that determines the model size k can be specified externally or estimated using finite population sampling theory. For this estimation, Principal Component Analysis (PCA) is employed, as detailed in [IZHZ23].

SDOclust. Given that SDO constructs a representative model for the data to calculate an outlier score, it is logical to leverage this model for clustering as well. SDOclust, as introduced by Iglesias et al. [IZHZ23], extends the SDO family of algorithms with this capability.

SDOclust enhances the algorithm by representing observers as nodes in an undirected graph, which undergoes a connected components clustering (CCC) process [HS00]. A local cutoff thresholding technique is employed to establish edges between these nodes. The connected components of the resulting graph are then interpreted as clusters within the model.

The cutoff threshold h_ω for a given observer ω is defined as follows:

$$h_\omega = d(\omega, \omega_{\leftarrow\chi}) \tag{3.1}$$

where $\omega_{\leftarrow\chi}$ denotes the χth-closest observer to ω, and $d(\cdot)$ represents the distance function. Then two observers ν, ω are connected if their distance is below both their local cutoff thresholds. In terms of the adjacency matrix A of the graph, this is expressed as:

$$A_{\nu,\omega} = \begin{cases} 1 & \text{if } d(\nu,\omega) < h_\omega \text{ and } d(\nu,\omega) < h_\nu, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

In Fig. 3.2, the mechanism of this concept is illustrated, demonstrating its effectiveness on non-convex clusters and shapes.

The local cutoff thresholding approach enables the algorithm to identify solutions of varying densities. However, this method also introduces the risk of connecting closely located clusters or creating clusters of noise. To mitigate this, a global density threshold h for the model is computed as the average of the local density thresholds. Subsequently, a mixture model is applied to determine the final cutoff threshold of an observer ω:

$$h'_\omega = \zeta \cdot h_\omega + (1 - \zeta) \cdot h, \tag{3.3}$$

where ζ is a mixing parameter.

After the graph is established each isolated subgraph is given a unique label c. Subsequently, each observer ω is assigned its own label $l_\omega \in C$ where C is the set of unique labels $C = \{c_1, c_2, \ldots, c_z\}$.

In a final step, the model is refined by removing isolated observers, typically originating from noisy areas and forming small subgraphs. Subsequently, observers from subgraphs with fewer than e nodes are removed from the model.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

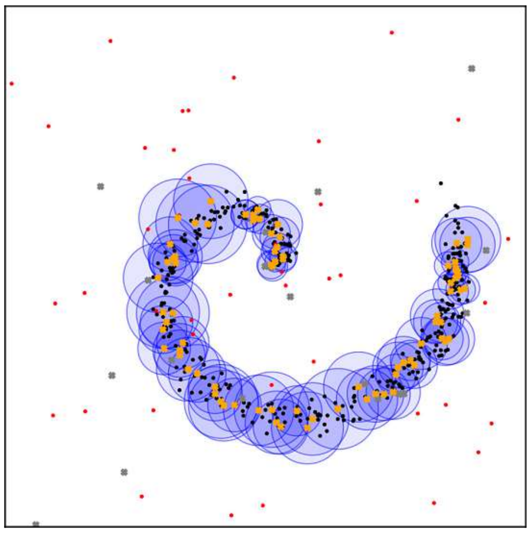


Figure 3.2: Illustration of a 2D data cluster shaped like a spiral with added noise (red dots, 10 percent noisy points). The figure shows the connectedness of points, referred to as observers, marked with 'x' (orange if active, grey if inactive). Each observer has a circle drawn to encompass its χ-th closest observer ($\chi = 4$). Overlapping circles between two active observers indicate that they share an edge in the graph, i.e., they are connected.

Prediction now operates similarly to SDO, but instead of assigning an outlier score, it assigns a label to each processed data point. The x nearest active observers to the data point s are identified, with each observer associated with a label $l \in C$. By counting the occurrences of each label among these x nearest active observers, the most frequent label or a membership vector is determined and returned.

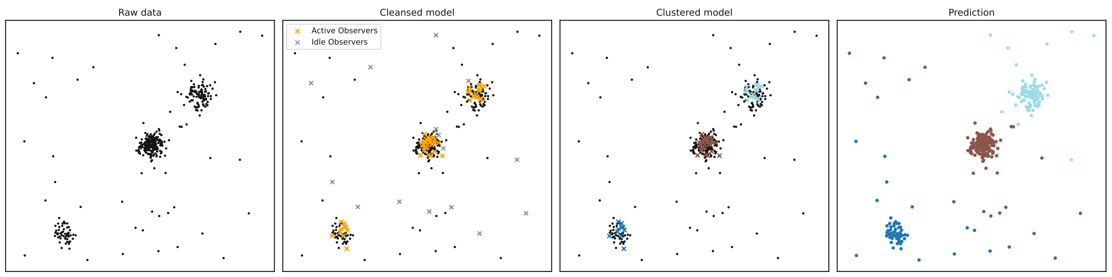


Figure 3.3: Illustration of the SDOclust algorithm, showcasing the key steps involved. Labels are color-coded for the clustered model and the predictions. Note that no outlier handling is embedded, so outliers are labeled with a best fit here.

The key steps of the clustering variation of the SDO family of algorithms are illustrated in Fig. 3.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

SDOstream. The SDO algorithm excels at naturally processing streaming data by using a fixed-size dataset as its model. A key challenge in enhancing this approach is adjusting the model to integrate new incoming data. To address this challenge, SDOstream [HIZ20] employs a damped time window strategy that focuses on a limited set of points rather than all past observations, ensuring the model remains up-to-date and accurate over time.

Algorithm 3.1: SDOstream

```
   Input: Data point $(v_i, t_i)$
1  Set $\omega_0 \leftarrow \text{argmin}_{\omega\in\Omega} \overline{P}_\omega$
2  Set $H_\omega \leftarrow f^{(t_i - t_{i-1})} H_\omega + 1$ and $P_\omega \leftarrow f^{(t_i - t_{i-1})} P_\omega \quad \forall \omega \in \Omega$
3  Set $P_\omega \leftarrow P_\omega + 1 \quad \forall \omega \in \mathcal{N}$
4  if $\Omega$ empty or $r \leq -k \cdot \ln(f)$ with $\in \mathbb{R}[0,1]$ then
5      if $|\Omega| = k$ then
6          Remove $\omega_0$ from $\Omega$
7      end
8      Add $v_i$ to $\Omega$
9      Set $P_{v_i} \leftarrow 1$, and $H_{v_i} \leftarrow 1$
10 end
11
```

To achieve this, SDOstream introduces a fading parameter $f \in (0, 1]$. This parameter controls the rate at which the model adjusts to new data, ensuring the model remains responsive and accurate over time. To make the fading parameter more intuitive, it is typically expressed as $f = \exp(-T^{-1})$, where $T \in \mathbb{R}^+$. This formulation allows T to be interpreted similarly to the window size in a sliding time window approach, providing a clearer understanding of how quickly old observations are forgotten.

Similar to the SDO algorithm, the observations P_ω for an observer ω serve as a quality metric that determines the idle-active split. This value P_ω is updated using an exponential moving average approach to adapt to new data. For each processed data point, P_ω is updated as follows: $P_\omega \leftarrow f \cdot P_\omega + 1$ if ω is among the x-closest observers to the data point, and $P_\omega \leftarrow f \cdot P_\omega$ if it is not.

Since SDOstream is operating with a fixed-size model and must continuously incorporate new data, it is essential to regularly replace observers with new data points. A quality metric like P_ω should be used for this task. However, a downside of this metric is that new observers would be constantly replaced, as they inherently have a lower P_ω compared to older ones. To address this, a normalized quality metric $\overline{P}_\omega$ is introduced, which adjusts P_ω in relation to its maximum possible value, here called the observers age H_ω. This maximum possible value can either be updated incrementally for each observer ω or approximated assuming a constant inter-arrival time of new data points, as detailed in [HIZ20].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Similar to the SDO algorithm, new data points are sampled randomly in SDOstream. To align with the fading parameter f, SDOstream employs a sampling rate designed so that each observer, on average, is replaced once every time $T = \frac{-1}{\ln(f)}$. This corresponds to a sampling rate of $\frac{k}{T} = -k \cdot \ln(f)$ where k is the number of observers. In contrast to the SDO algorithm, idle observers in SDOstream should not be removed. Instead, they are retained as they may become active over time.

The detailed framework for LEARNING in the algorithm is outlined in Algorithm 11. The PREDICTION process remains unchanged to SDO. Compute the median distance between the processed data point and its x nearest active observers using the currently active model.

In SDOstream, it is important to note that the data stream is unbounded, preventing the use of finite populations sampling theory as is done in SDO and SDOclust. Consequently, the model size must be specified as an external parameter, relying on the user's knowledge of the domain to set an appropriate value.

3.1.2 Core algorithm of SDOstreamclust

The primary challenge, compared to SDOstream, lies in the continuous maintenance and dynamic updating of the graph representation from SDOclust to account for an evolving set of active observers as the data stream progresses. This difficulty is particularly pronounced due to the definition of the cutoff threshold (as described in equation (3.1)), which requires determining the χth-closest observer, denoted as $\omega_{\leftarrow\chi}$, for each observer $\omega \in \Omega_a$. Changes in the set of active observers can alter the χth-closest observer for a given ω, leading to subsequent changes in both the global density threshold h and the entire graph representation, as defined by the corresponding adjacency matrix in equation (3.2). Algorithm 3.2 outlines the primary supplementary steps integral to the SDOstreamclust framework.

Algorithm 3.2: SDOstreamclust

```
  Input: Data point $(v_i, t_i)$
1 SDOstream
2 $\mathcal{C} \leftarrow$ Cluster
3 $\mathcal{L} \leftarrow$ Label
4 Set $L^c_\omega \leftarrow f^{(t_i - t_{i-1})} L^c_\omega \quad \forall \omega \in \Omega, \forall c \in C$
5 Set $L^c_\omega \leftarrow L^c_\omega + 1 \quad \forall \omega \in \mathcal{L}_c$
```

The LEARNING phase begins, using the same fitting procedure as in SDOstream (refer to Algorithm 11). Subsequently, the additional steps unique to SDOstreamclust are executed. In **line 2**, the current cluster representation is constructed, as detailed in Algorithm 3.3. Once the cluster representations are established, they undergo a labeling process, which is performed in **line 3**, detailed in Algorithm 3.5. A comprehensive discussion of those two steps is provided in Section 3.1.3. In the final step of the LEARNING phase (**lines**

4-5), the label affiliation statistics are updated. These statistics are essential for both the PREDICTION phase and the assignment of labels to clusters within the model. This process starts with applying the predefined fading mechanism, which gradually reduces the influence of older data, and is followed by updating the statistics based on the labels observed in the current set $\mathcal{L}$.

In the PREDICTION phase, a cluster label is assigned to the processed data point. Unlike SDOclust, where each observer ω is associated with a unique label $l_\omega \in C$, in this case, each observer is linked to a cluster-label membership vector $\mathbf{l}_\omega \in \mathbb{R}^{|C|}$, derived from previous cluster observations L_ω. The membership vector represents the affiliation probabilities across all known clusters or classes. Consequently, the most dominant cluster label among the x nearest observers is selected and assigned to the data point, as determined by:

$$l_i \leftarrow \underset{c \in C}{\operatorname{argmax}} \sum_{\omega \in \mathcal{N}_a} \mathbf{l}_\omega \tag{3.4}$$

The data structure to be maintained consists of k observers, denoted as Ω, which are continuously replaced over time. Each observer ω is characterized by its position (determined by the original data point from which it was sampled), observations P_ω, age H_ω, and cluster-observations $L_\omega \in \mathbb{R}^{|C|}$. Throughout the process, P_ω, H_ω, and L_ω are updated incrementally. The cluster-observations L_ω represent a significant enhancement over SDOstream, as they facilitate the retention of historical statistics crucial for accurate temporal propagation and the inheritance of labels.

3.1.3 Cluster Formation and Labeling

In the model, a cluster corresponds to a connected component in the graph with at least e nodes, as discussed in Section 3.1.1. Consequently, a depth-first search (DFS)-inspired approach, as outlined in Algorithms 3.3 and 3.4, is a natural choice. This method efficiently gathers observers that constitute a connected component, thereby representing an individual cluster within the graph.

Algorithm 3.3: Cluster

```
  Output: Set of clusters $\mathcal{C}$
1 for $\omega \in \Omega_a$ do
2     if $\omega$ not visited then
3         $S \leftarrow$ DFS[$\emptyset, \omega$]
4         if $|S| \geq e$ then add $S$ to $\mathcal{C}$ end
5     end
6 end
7 return $\mathcal{C}$
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm 3.4: DFS

```
   Input: Cluster $S$, Observer $\omega$
   Output: Cluster $S$
1  Add $\omega$ to $S$. Mark $\omega$ as visited
2  for $\nu \in \Omega_a$ do
3      if $\nu\omega$ is edge and $\nu$ not visited then
4          $S \leftarrow \texttt{DFS}[S, \nu]$
5      end
6  end
7  return $S$
```

Since the model (i.e., observers) evolves over time, the graph also undergoes changes. Consequently, determining whether clusters identified at different time points correspond to the same cluster is a non-trivial challenge. If the algorithm is robust — meaning the model parameters are chosen appropriately — observers are unlikely to frequently change their cluster affiliation. However, the algorithm must also be capable of handling novelty and change, particularly in relation to the scenarios involving concept drift described in Section 2.3.1. This ability to adapt is crucial to maintaining accuracy as the model evolves over time.

The fundamental assumption in labeling clusters over time during stream processing is that the labeling of observers remains stable and robust. Therefore, if observers are not replaced too rapidly, it is reasonable to assign a cluster's label based on a majority vote reflecting their past affiliations. However, challenges arise in scenarios where two clusters vote for the same label or when one cluster is absorbed by a larger cluster—situations commonly referred to as split and merge scenarios. Additionally, new clusters may emerge over time.

Addressing these challenges requires careful analysis to ensure accurate and adaptive labeling in dynamic environments. In the case of a split, it is important to determine whether the new split genuinely represents a robust class or merely reflects model uncertainty. Conversely, in a merge scenario, it is essential to verify that the merging of clusters is not a temporary effect caused by noisy data, and that the original cluster is not prematurely discarded. In general, when new clusters emerge, whether through splitting or as entirely new formations, it is crucial to confirm that these clusters represent actual robust classes rather than mere outliers.

To accurately track corresponding clusters over time, it is crucial to maintain statistics on the affiliation of observers with specific clusters. Similar to the principle where a data point is monitored by its x nearest observers, a cluster is considered observed by an observer if the observer belongs to that cluster at the time of observation. As before, past observations gradually diminish in influence. For each observer ω, a record of cluster observations, denoted as L_ω, is maintained. These observations are then utilized both for labeling new cluster models — represented by the graph at a given time — and for

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

labeling data points during stream processing.

As a strategy for assigning labels to a new cluster model, priority should be given to clusters that are clearly defined by a majority of their observers' past label affiliations, using as much historical information as possible. This approach is particularly critical in split scenarios, where the larger split should inherit the established label. The smaller split may be considered a candidate for a novel class, though this possibility requires future validation. It is essential to ensure that the robustly established class continues to be represented within the larger split, maintaining continuity and accuracy in labeling.

Consequently, we establish the following procedure for assigning labels to a set of cluster representations $\mathcal{C} = \{S_1, S_2, \dots, S_n\}$, where each cluster representation S_i is a set of observers, and for each observer ω, its cluster observations $L_\omega \in \mathbb{R}^{|C|}$ are known. For each cluster representation S, we calculate a score $\bar{s}$ and determine a candidate label $\bar{c}$ as follows:

$$\bar{s} = \max_{c \text{ available}} \sum_{\omega \in S} \mathbf{l}_\omega^c \quad \text{and} \quad \bar{c} = \operatorname*{argmax}_{c \text{ available}} \sum_{\omega \in S} \mathbf{l}_\omega^c \tag{3.5}$$

Here, $\mathbf{l}_\omega$ denotes the cluster membership vector, i.e., the cluster observations L_ω are normalized so that their sum equals 1. This normalization ensures that each observer contributes equally to the label assignment, regardless of their age. The score s determines the priority queue $\mathcal{Q}$ for assigning labels to the cluster representations S. Each cluster must receive a unique label, meaning that once a label is assigned to one cluster, it cannot be used for another cluster at this time anymore.

Algorithm 3.5: Label

```
   Input: Clusters $\mathcal{C}$
   Output: Labeled clusters $\mathcal{L}$
 1 Set priority queue $\mathcal{Q}$ from $\mathcal{C}$ using criterion (3.5)
 2 for $S \in \mathcal{Q}$ do
 3     Drop $S$ from $\mathcal{Q}$
 4     if $\bar{s} = 0$ then
 5         Set novel labeled cluster $\mathcal{L}_{|C|+1} \leftarrow S$
 6         continue
 7     end
 8     if $\bar{c}$ is available then
 9         Set labeled cluster $\mathcal{L}_{\bar{c}} \leftarrow S$ and mark $\bar{c}$ as taken
10     else
11         Insert $S$ to $\mathcal{Q}$ using recalculated criterion (3.5)
12     end
13 end
14 return $\mathcal{L}$
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In the if-statement beginning at **line 4**, a new candidate class emerges within the model. The condition $\bar{s} = 0$ signifies that all previously recognized classes are already associated with other clusters where they are more prominent. A new class appears only under these conditions. Although the class is added to the model, it is not yet guaranteed that any data point will be assigned to it. Such assignments generally occur when the class stabilizes within the model and the cluster gradually establishes itself as distinct over time.

The if-else statement starting at **line 8** either assigns a label to the cluster if the candidate label $\bar{c}$ is still available, or it triggers a recalculation. This involves updating the priority queue and proceeding to the next cluster based on the priority.

3.1.4 Nearest-neighbor search

A frequent subtask within the algorithm is performing nearest-neighbor searches in the model. This is critical for identifying the x-nearest observer sets, $\mathcal{N}$ and $\mathcal{N}_a$, as well as determining the cutoff threshold h_ω for an observer ω, as defined in (3.1). In particular, it is necessary to find the χth closest observer, $\omega_{\leftarrow\chi}$, for each observer ω.

Another related task involves identifying all observers within a specific range when searching for observers ν that have an edge with ω (see Algorithm 3.4, line 3). Generally, the complete graph or its adjacency matrix is not fully known or necessary. Therefore, candidate observers ν are those that are closer to ω than the threshold h'_ω.

This task must be performed repeatedly in a dynamic environment, where the model frequently undergoes changes. With each processed point, the set of active observers can change due to the addition of new observers, the removal of old ones, or transitions between active and inactive states. To address these requirements, we utilize a data structure that supports efficient x-nearest neighbor searches, range queries and incremental updates, including insertions, deletions, and replacements.

Hartl [Har23] recommends using M-trees [CPZ+97] for this purpose. M-trees are particularly well-suited for streaming scenarios because they allow for the efficient updating of an existing tree structure. This spatial indexing data structure facilitates efficient nearest neighbor and range queries within metric spaces. By leveraging M-trees, we can dynamically adjust the tree structure after its initial construction, ensuring efficient updates and adaptability in a streaming context.

3.1.5 Outlier thresholding

SDOstreamclust provides an outlierness score by default (SDOstream [HIZ20]) but generally is not built to handle outlier thresholding internally. A wide range of methods are available for this task. For example, in [HHH+22], Han et al. compare 30 different techniques for identifying outlier thresholds in one-dimensional data, all of which are available in the Python package PyThresh[1].

[1] `https://github.com/KulikDM/pythresh`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

While external thresholding is recommended, an option to estimate thresholds internally is offered. In this approach, an activation function $g : \mathbb{R}^+ \to [0, 1)$ based on the hyperbolic tangent is applied to transform the distances $d(v_i, \omega)$ between data points and observers into probabilistic outlier scores i.e.

$$p_i^o(\omega) = \begin{cases} g\left(\lambda \cdot \frac{d(v_i,\omega) - h'_\omega}{h'_\omega}\right) & h'_\omega \leq d(v_i, \omega) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

The underlying intuition is that if a data point lies closer to an observer than the observer's cutoff threshold, it is likely to belong to the same cluster as the observer, leading to a zero probability of being classified as an outlier with respect to that observer. As the distance between the data point and the observer increases relative to the observer's cutoff threshold—representing the local density around the observer—the likelihood of the point being classified as an outlier also increases proportionally. In Fig. 3.4, this mechanism is demonstrated using the same setup as in Fig. 3.2.

The parameter h^o denotes the number of cutoff thresholds h'_ω by which a data point must be separated from its nearest observers to reach a 50 percent probability of being classified as an outlier. Thus, the scaling factor λ for the activation function $g(\cdot)$ is determined by the equation:

$$g\left(\lambda \cdot (h^o - 1)\right) = 0.5 \tag{3.7}$$

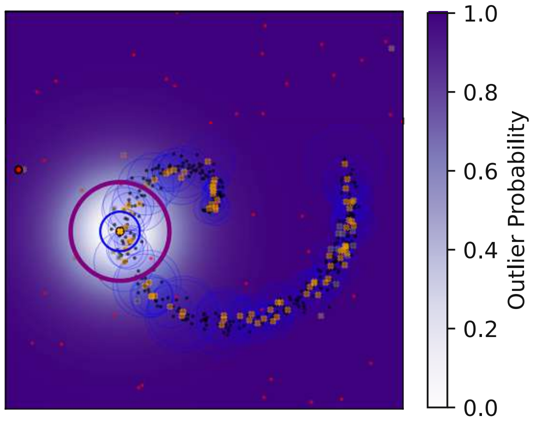


Figure 3.4: 2D data cluster similar to Fig. 3.2. The red dot with a black edge indicates a processed point, likely an outlier detected by human judgment. The radial color scale represents outlier probability relative to a highlighted close observer. The blue circle marks the cutoff threshold ($\chi = 4$), and the purple circle indicates the 50% outlier threshold, defined by the relative outlier sensitivity parameter $h^o = 2.5$.

A data point is classified as an outlier and its cluster label is replaced with the outlier label if the average probabilistic outlier score among its nearest active observers, $\mathbf{p}_i^o$, exceeds 0.5. This average score is computed as follows:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

$$\mathbf{p}_i^o \leftarrow \underset{\omega \in \mathcal{N}_a}{\text{mean}}\, p_i^o(\omega) \tag{3.8}$$

The probabilistic outlier scores $p_i^o(\omega)$ can also be interpreted as a measure of how well an observer ω fits the data point v_i it is observing. Under this interpretation, the PREDICTION step, described by equation (3.4), can be adapted to a weighted approach. In this approach, the cluster label l_i is determined by:

$$l_i \leftarrow \underset{c \in C}{\text{argmax}} \sum_{\omega \in \mathcal{N}_a} (1 - p_i^o(\omega)) \cdot \mathbf{l}_\omega \tag{3.9}$$

Here, $(1 - p_i^o(\omega))$ represents the fit of observer ω to the data point, and $\mathbf{l}_\omega$ denotes the cluster label associated with observer ω.

As a final remark, while here the hyperbolic tangent function is employed as the activation function, it is worth noting that alternative activation functions may also be employed.

3.1.6 Batch processing and input buffer

Although streaming data analysis typically relies on incremental processing, batch operations can sometimes be advantageous. Batches provide more comprehensive contextual information for analysis, and in many cases, applications can tolerate batch processing effectively.

Batch processing is well-suited to the proposed algorithm, requiring only minor adjustments to the usual sequence of operations. Typically, the process follows the order of prediction, fitting, and then sampling or replacement. In the batch processing framework, however, sampling or replacement occurs first, necessitating a queue based on the criterion $\overline{P}_\omega$ when multiple observers are to be replaced. This is followed by model fitting and then the prediction phase. Special attention is required in these latter steps to properly handle nearest neighbor calculations for points added to the model through sampling. Specifically, when determining $\mathcal{N}, \mathcal{N}_a$, or the χ-closest observer, the newly sampled points must be excluded from the computations.

Batch processing enhances efficiency by minimizing the frequency of computationally intensive graph calculations, which are executed once per batch rather than for each individual data point. Similarly, an input buffer can be utilized to further optimize performance. In this approach, the LEARNING phase is postponed until the buffer reaches its capacity, at which point the learning process is carried out in a single iteration, creating a virtual batch. As long as the buffer size remains small relative to the dynamics of the data stream, its impact on overall performance is minimal. The buffer's main role is to reduce computational load, which, in the case of SDOstreamclust, is primarily focused on clustering Ω_a.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Cluster observations during batch processing must be slightly modified to account for batch age for fair scoring. The batch age is defined similarly to observer age and is updated iteratively:

$$H_B = \begin{cases} 1 & \text{if } i = 1 \\ f^{(t_{b_i} - t_{b_{i-1}})} H_B + 1 & \text{if } i > 1 \end{cases} \tag{3.10}$$

where $B = \{(v_{b_1}, t_{b_1}), (v_{b_2}, t_{b_2}), \ldots, (v_{b_n}, t_{b_n})\}$ denotes the batch. The update is given by:

$$\begin{aligned} L_\omega^c &\leftarrow f^{(t_{B_i} - t_{B_{i-1}})} L_\omega^c \quad \forall \omega \in \Omega, \forall c \in C \\ L_\omega^c &\leftarrow L_\omega^c + H_B \quad \forall c \in \mathcal{L} \end{aligned} \tag{3.11}$$

where t_{B_i} denotes the timestamp of the ith processed batch.

3.1.7 Parameters

Most parameters are inherited from SDO, SDOstream, and SDOclust, as SDOstreamclust combines and enhances these approaches. A broader discussion can be found in [VZZ18], [HIZ20], and [IZHZ23]. Below, we categorize and discuss SDOstreamclust parameters based on their characteristics:

- *Rate of Change:* Two key parameters in this category are the number of observers k and the number of neighbors x, both of which influence the model's responsiveness to data changes. The number of observers k and the idle-observer threshold ρ together define the size of the active model, approximately $(1 - \rho) \cdot k$, directly affecting its granularity in capturing relevant data patterns. However, there is a trade-off: a larger model becomes more rigid to changes, while a smaller model can adapt more readily.

  The number of neighbors x determines which aspects of the model gain importance when processing each data point. A small x can lead to overfitting, making the model overly sensitive to fluctuations, whereas a high x may oversimplify the model, resulting in reduced stability and a less accurate representation of the underlying data patterns.

- *Temporal memory:* The fading parameter $f \in (0, 1)$ controls how quickly the model adapts to new data clusters, while also influencing its stability in the presence of noise. For ease of interpretation, user input is provided as a time parameter $T \in \mathbb{R}^+$. This parameter functions similarly to the window size in sliding window (SW) approaches, with T controlling the model's sensitivity to changes in the data. The relationship between f and T is defined as $f = \exp(-T^{-1})$. The choice of T is highly dependent on the application.

  Additionally, this parameter influences the *rate of change* in cluster positions, controlling how quickly clusters can shift spatially in response to new data.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *Cluster formation:* The parameter χ defines the local cutoff thresholds for observers in the model. In our implementation, χ is expressed either as a fraction of the number of observers k or as a minimum value. Intuitively, χ should be set lower when a greater number of distinct clusters is expected, and higher when fewer, larger clusters are preferred. However, caution is needed: lower χ values may lead to overfitting by creating too many small clusters, while higher values risk underfitting, failing to capture meaningful distinctions between clusters. The optimal χ depends heavily on the specific application and dataset.

  The parameter ζ controls the trade-off between local and global thresholding when cutting graph edges: $\zeta = 1$ corresponds to purely local thresholding, and $\zeta = 0$ to purely global thresholding. Local thresholding accommodates clusters with significant density differences but risks merging nearby clusters and forming clusters in noisy areas. In contrast, global thresholding avoids these mergers but may split legitimate clusters.

  The parameter ρ cleans the model from outliers and noise. A low ρ risks including noise and outliers, while a higher ρ may oversimplify the model, potentially reducing its accuracy. Thus, ρ must strike a balance between computational efficiency and the model's ability to accurately represent the data.

  The parameter e (minimum observers) prevents noise or outliers from being misclassified as clusters, improving robustness. However, if set too high, it risks overgeneralization.

- *Outlier thresholding:* The relative outlier sensitivity parameter h^0 scales the activation function used to determine the outlier probability. Higher values of h^0 result in fewer points marked as outliers, while lower values increase the rate of marked outliers.

We anticipate that the parameters for algorithms within the SDO family will demonstrate considerable robustness. This means that the same parameterization can tolerate multiple different scenarios while maintaining optimal accuracy, and also that the parameters tolerate ranges of variation without affecting the accuracy obtained for a given scenario. This has been convincingly evaluated in SDOclust [IZHZ23] and recently for the parameter x in SDO [IMZ24], which stands out as the most stable neighborhood parameter-dependent anomaly detection algorithm among those evaluated in the cited work. However, experiments will be conducted to evaluate their performance, with a more detailed analysis and discussion provided in Section 4.2.

3.1.8 Initialization and Data Integrity

Model initialization. The warm-up phase is a critical initial period in the lifecycle of a model, where it transitions from an untrained or partially trained state to a more robust and fully operational status. During this phase, several challenges and issues may arise that can significantly impact the model's performance and stability.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

One of the primary concerns during the warm-up phase is model instability. Since the model is not yet fully trained, its predictions may be unreliable and subject to high variance. This instability can lead to inconsistent outputs and make it difficult to gauge the model's effectiveness accurately.

Another challenge is inadequate learning. In the early stages, the model may not yet have enough data or experience to capture the underlying patterns in the data effectively. This can result in poor performance, as the model struggles to learn meaningful representations from the limited information available.

An essential adaptation for our method during the warm-up phase is the proportional adjustment of parameters related to the observer size k. Specifically, adjustments may be necessary for parameters such as the number of neighbors x, the local threshold for graph cutting χ, and the minimum number of observers e. These adjustments ensure that the method remains effective and robust as the model evolves from its initial, less stable state.

In the early phase of model training, points that have been sampled and observed frequently tend to have an advantage due to the smaller model size (i.e., the number of observers). Assuming randomness in data sampling, the probability of a given observer having observed a particular data point is given by $\frac{x}{k} \cdot \frac{x-1}{k-1} \cdots \frac{2}{k-x+2} \cdot \frac{1}{k-x+1}$. This expression represents the reciprocal of the binomial coefficient $\binom{k}{x}$, which denotes the number of ways to choose x points from a total of k observers. This probability reflects the likelihood of a data point being observed by a specific number of observers during the initial stages of model development when the model size is still small.

This adjustment simplifies the observation scoring during the warm-up phase, as outlined in lines 2-3 of Algorithm 11. The updated scoring formula is:

$$P_\omega \leftarrow P_\omega + \left[\binom{\tilde{k}}{\tilde{x}} \div \binom{k}{x}\right] \quad \forall \omega \in \Omega$$

Here, the parameters with tildes ($\tilde{k}$ and $\tilde{x}$) are those used at this time during the warm-up phase, while k and x are the target parameters of the final model. Similarly, the age H_ω is updated in the same manner as P_ω.

Duplicate points. A challenge observed in stream clustering is the frequent recurrence of identical data points, which can disrupt the clustering process. This issue arises because only a limited number of observers, denoted as x, can evaluate a given data point. When more than x observers are assigned to the same point, it becomes unclear which observer should score the observation, potentially leading to temporal bias.

A more elegant solution is to augment the data point by adding a small noise term in an additional dimension. Our implementation provides this functionality. The noise term ϵ

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

should be chosen such that $d(v_i, v_j) > \epsilon$ for distinct data points $v_i \neq v_j$, ensuring that identical points remain distinguishable and mitigating the bias introduced by duplicates.

This procedure is implemented but is disabled by default; however, it is highly recommended if the dataset is known to contain duplicate points.

3.2 Evaluation

This section evaluates SDOstreamclust, covering datasets, benchmark algorithms, parameter tuning, and performance metrics. All materials and experimental setups are available in a DOI-citable repository [IV24] and can be replicated using Docker[2]. SDOstreamclust can also be installed via `pip` [3] or directly from the repository[4].

3.2.1 Datasets

This section presents the datasets used, featuring both synthetic data and real-world applications.

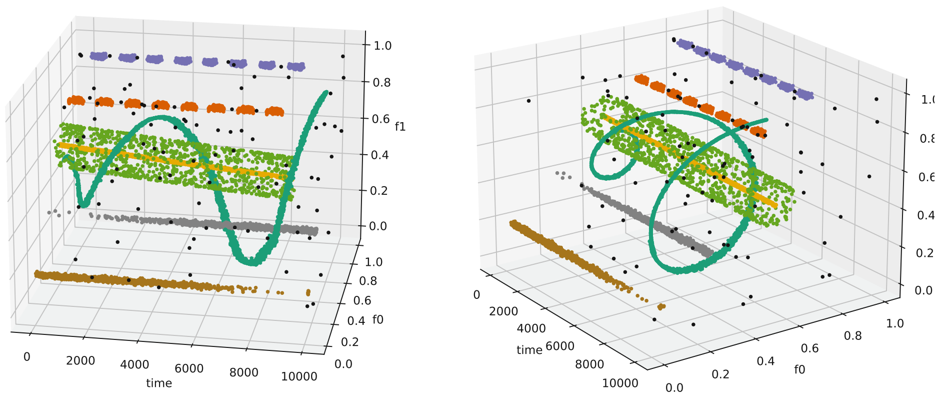


Figure 3.5: The Drifting Conglomerates dataset. Referring to Fig. 2.3, the blue and orange clusters alternate in frequency, exemplifying sudden and reoccurring drift. The brown and grey clusters display gradual drift, with brown often preceding a slow replacement by grey. The dark green cluster follows a spiral pattern, simulating incremental drift while exhibiting a non-convex shape. The light green and yellow clusters are geometrically concentric with significantly different densities, moving linearly and further demonstrating incremental drift while maintaining geometric alignment, which presents additional challenges due to their proximity and varying density. Outliers are marked in black.

[2]`https://hub.docker.com/r/fiv5/sdostreamclust`
[3]`https://pypi.org/project/pysdoclust-stream/`
[4]`https://github.com/CN-TU/pysdoclust-stream/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Drifting Conglomerates. The synthetic dataset presents challenges for stream clustering algorithms due to its complex patterns, including simultaneous concept drift, outliers, noise, non-convex shapes, and varying densities. In two dimensions, it comprises approximately 10,000 data points across seven types of evolving clusters, serving as a benchmark for evaluating clustering algorithms' versatility regarding these complex patterns. The data is illustrated in Fig. 3.5, color-coded by label.

Specific concept drift. This collection of datasets contains 160 scenarios generated using MDCstream [IOHZ20], available for download from the repository [Igl21]. Each scenario contains 10,000 data points with dimensionality ranging from 3 to 30 and 2 to 10 clusters that vary in size and spatial distribution. The categories are as follows:

a. *base* refers to the absence of concept drift, characterized by stationary clusters that appear with consistent frequency,

b. *sequential* simulates sudden drift, where clusters appear one by one over time, meaning that no two clusters appear simultaneously,

c. *non-stationary* simulates both sudden and gradual drifts, as well as reoccurring contexts. In this type, clusters coexist, appear, and disappear in an arbitrary manner, and

d. *moving* involves all clusters moving linearly through space at different speeds, representing incremental drift.

All datasets in these categories have both a *clean* version, where outliers are explicitly excluded, and a *normal* version, which includes outliers at a rate of less than 5%. Fig. 3.6 illustrates this dataset and its categories using two dimensions and time for visualization.

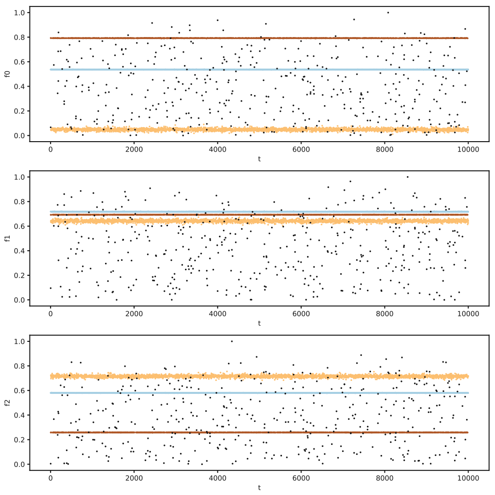
Real-world datasets. This collection features four datasets originating from diverse real-world applications, each exhibiting time dependency.

a. *Network-traffic flows* with 22 numerical features and timestamps, extracted from the first 10,000 entries of the TII-SSRC-23 dataset. It is used to evaluate Intrusion Detection Systems [HLA23], with Ground Truth defined by the `subtypo` attribute across 6 classes.

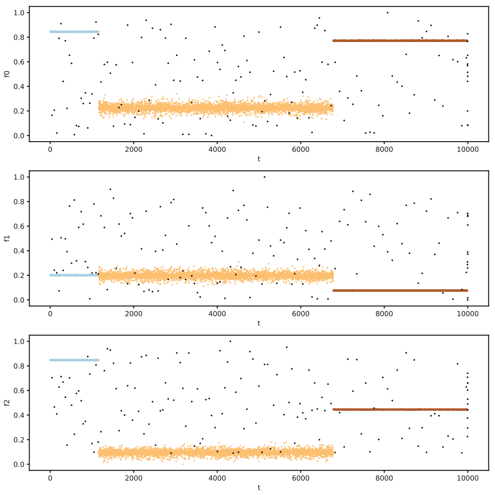
b. *Occupancy.* The dataset [CF16] includes temperature, humidity, light, CO2 concentration, and humidity ratio to predict room occupancy (`occupied` or `empty`). It consists of around 8,000 entries recorded every 5 minutes, with timestamps provided.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

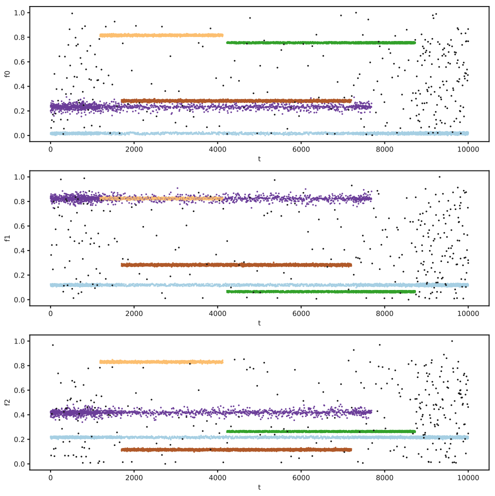
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



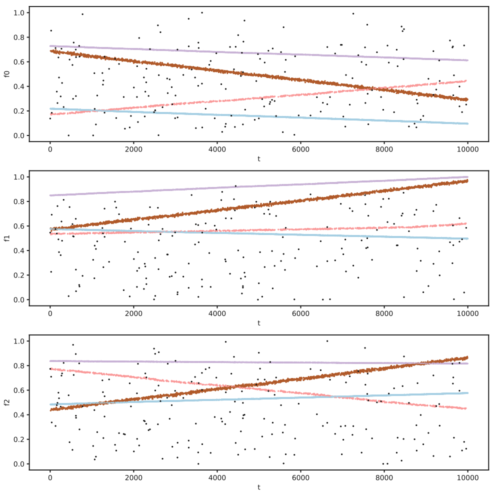
(a) Base



(b) Sequential

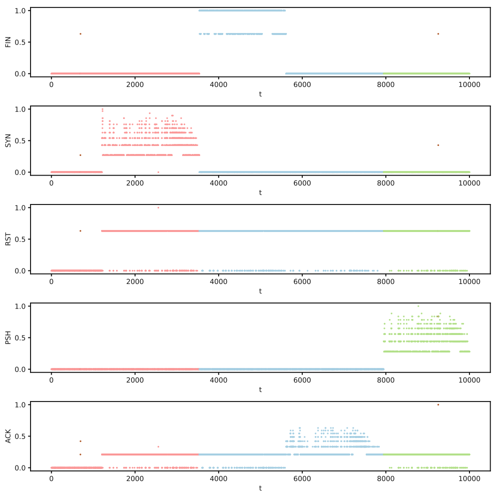


(c) Non-stationary

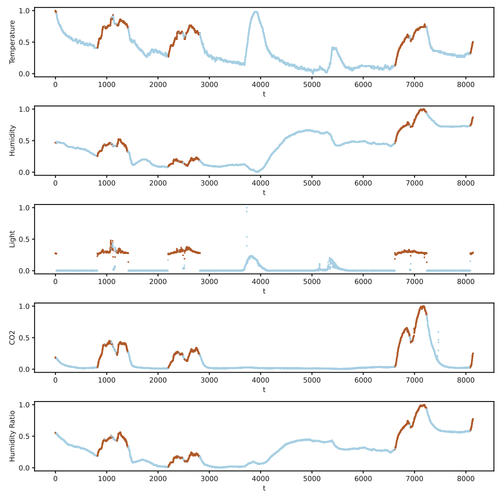


(d) Moving

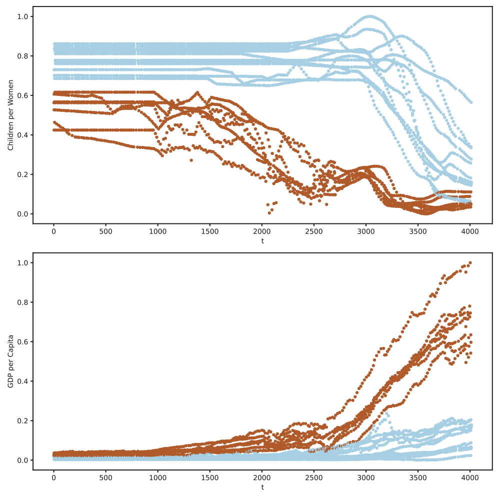
Figure 3.6: Examples of the Specific Concept Drift datasets. Clustered data is shown colored while outliers in black. Only three dimensions and time are shown.



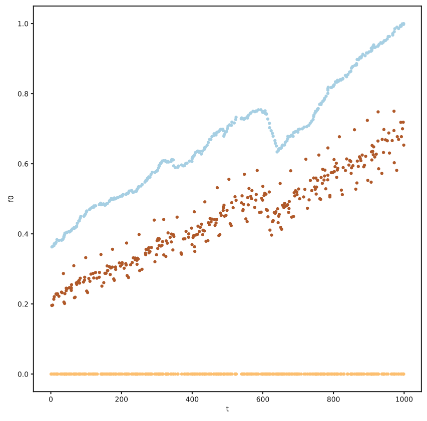
(a) Network-traffic (22 dims.)



(b) Occupancy (5 dims.)



(c) Fertility vs Income (2 dims.)



(d) Retail (1 dim.)

Figure 3.7: Plots of the Real-world Datasets collection. Colored according to the clusters defined in the Ground Truth.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

c. *Fertility vs Income.* This dataset, available from Gapminder [5] and through version from [IVZ23], includes average babies per woman and GDP per capita (in PPP dollars) for 18 countries, with a binary ground truth label: `European` or `non-European`.

d. Available from the U.S. Census Bureau and the Federal Reserve Economic Database (FRED) [6], the *Retail* dataset, as used in [IVZ23], includes a combined time series of RETAILIMSA, RETAILIRSA, and RETAILSMNSA with approximately 1,000 data points and three distinct classes.

Fig. 3.7 visually illustrates four real-world datasets, providing insight into the challenges associated with drift, density, and geometry.

3.2.2 Algorithms and Parameters

All algorithm implementations used in the evaluation, except for SDOstreamclust, are from the River package [7]. Those are

- *CluStream* [APHW03] maintains statistical information through micro-clusters, which store temporal data summaries following a pyramidal pattern. When a new point arrives, it is either added to the closest micro-cluster or, if it doesn't fit, space is freed by deleting or merging clusters. This improved version uses Welford's algorithm for incremental variance calculation. Since River lacks an offline phase, a timegap parameter triggers incremental K-Means clustering on the micro-clusters to form macro-clusters.

- *DBStream* [HB16] is a stream clustering algorithm that tracks the density between micro-clusters using a shared density graph. When a new data point arrives, the algorithm either creates a new micro-cluster or updates nearby clusters by adjusting their weights and positions relative to the point. The shared density graph is updated accordingly, and weak clusters are periodically removed to optimize performance. For offline clustering, a connectivity graph is built from strong micro-clusters, and a variant of DBSCAN [EKS+96] is used to form macro-clusters based on density connections between the clusters.

- *DenStream* [CEQZ06] is a stream clustering algorithm that detects clusters with arbitrary shapes and is robust against noise. It uses core-micro-clusters to summarize dense areas and employs a pruning strategy to manage potential and outlier micro-clusters, ensuring memory efficiency. When a new point arrives, it is either merged into an existing potential or outlier micro-cluster, or a new outlier cluster is created.

[5]`https://www.gapminder.org/data/`

[6]`https://www.kaggle.com/datasets/census/retail-and-retailers-sales-time-series-collection`

[7]`https://github.com/online-ml/river`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Periodically, micro-clusters with weights below a threshold are deleted. Offline, a variant of DBSCAN [EKS+96] connects density-based potential micro-clusters to form final clusters.

- *StreamKMeans* [OMM+02] uses an incremental K-Means approach. Initially, cluster centers are set using a K-Means instance. As new points arrive, they are added to a temporary chunk until it reaches its maximum size, at which point a new incremental K-Means instance processes the chunk, updating the centers. The chunk is then cleared for new points.

KMeans and *TextClust* were intentionally excluded from the evaluation, with *KMeans* not being incremental and *TextClust* specializing solely in text clustering. River [MHM+21] was chosen for its reliability, performance, and simplicity, providing efficient implementations of the key clustering algorithms widely recognized in the field.

Table 3.2 shows the hyperparameters set during experimentation for each algorithm. Each method presents a different constellation of them, but usually with similar functions that can be summarized in the following groups:

a. *number of clusters*, in those algorithms that need to be imputed with this value externally

b. *internal update*, referring to an internal batch or buffer for updating specific computations, models or routines

c. *rate of change*, indicating the inertia of clusters to vary their spatial position

d. *cluster formation*, defining cluster boundaries, composition, distances, and density thresholds

e. *outlier thresholding*, determining coefficients to disclose outliers

f. *temporal memory*, which sets how long and with what weight past information has to be retained, affecting also the speed of adaptation to new changes

As in static clustering, it is desirable that the hyperparameters be as minimal as possible, robust, intuitive and self-adjusting. However, in streaming data analysis, particularly the parameters related to the memory or temporal window are highly subjective. They determine whether the algorithm should remember for a long time or forget soon and become more flexible to change. Such parameters, yet intuitive and easy to tune, are commonly unavoidable and require the adjustment based on expert knowledge.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 3.2: Hyperparameters. Searched indicates values optimized through Grid Search; otherwise, they are fixed. Non-listed hyperparameters use default values.

	parameter	type	values	searched
CluStream	`time_gap`	internal update	ξ	no
	`n_macro_clusters`	number of clusters	κ	no[‡]
	`time_window`	temporal memory	$[10, \tau]$	yes
	`max_micro_clusters`	cluster formation	$[5 \cdot \kappa, 20 \cdot \kappa]$	yes
	`micro_cluster_r_factor`	cluster formation	[1.5, 4]	yes
	`sigma`	cluster formation	[0.1, 5.0]	yes
	`mu`	cluster formation	[0.0, 1.0]	yes
	`halflife`	rate of change	[0.1, 0.9]	yes
DBstream	`cleanup_interval`	internal update	ξ	no
	`clustering_threshold`	cluster formation	[0.1,...,3]	yes
	`fading_factor`	temporal memory	[0.01, 0.5]	yes
	`intersection_factor`	cluster formation	[0.1, 0.9]	yes
	`minimum_weight`	outlier thresholding	[0.1, 0.9]	yes
DenStream	`mu`	outlier thresholding	20	no
	`stream_speed`	internal update	ξ	no
	`max_micro_clusters`	cluster formation	$[5 \cdot \kappa, 20 \cdot \kappa]$	yes
	`decaying_factor`	temporal memory	[0.1, 0.9]	yes
	`beta`	outlier thresholding	[0.1, 0.9]	yes
	`epsilon`	cluster formation	[0.01, 0.3]	yes
streamKMeans	`chunk_size`	internal update	ξ	no
	`n_clusters`	number of clusters	κ	no[‡]
	`halflife`	rate of change	[0.1, 0.9]	yes
	`sigma`	cluster formation	[0.1, 5.0]	yes
	`mu`	cluster formation	[0.0, 1.0]	yes
SDOstreamclust	`input_buffer`	internal update	ξ	no
	`k`	rate of change	500[†]	no
	`outlier_threshold`	outlier thresholding	[2, 7]	yes
	`T`	temporal memory	$[\mathtt{k}, \tau]$	yes

κ: number of clusters, $\xi = 10$: internal update interval, $\tau = 0.1 \cdot n$: training size, n: data size.
Notation: $f \leftarrow \exp(-\mathtt{T}^{-1})$, $h^o \leftarrow$ **`outlier_threshold`**.

[†]: Small datasets (e.g., Retail, Fertility vs. Income) use `k`=100 observers.
[‡]: Clusters input; algorithms leverage Ground Truth (κ).

3.2.3 Validation metrics

Since the datasets used in the experiments include ground truth labels, a measure that assesses the degree of agreement between the algorithm's predictions and the ground truth labels is utilized. The Rand Index (RI) [Ran71] evaluates clustering by comparing pairs of elements to see if they are assigned to the same or different subsets in both partitions. Specifically,

$$\mathrm{RI} = \frac{a + b}{\binom{n}{2}} \tag{3.12}$$

where a is the number of pairs of elements assigned to the same subset in both partitions, b is the number of pairs of elements assigned to different subsets in both partitions, and $\binom{n}{2}$ is the total number of possible pairs of points.

The Rand Index (RI) has several downsides: it doesn't account for chance agreement, lacks normalization, and can give misleadingly high scores for random clustering. The

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Adjusted Rand Index (ARI) [HA85] addresses these issues by adjusting for chance, providing a normalized and more accurate measure of clustering quality. The Adjusted Rand Index (ARI) ranges from -1 to 1. A value of 1 indicates perfect agreement between the clusterings, 0 signifies performance equivalent to random chance, and negative values suggest that the clustering agreement is worse than what would be expected by chance.

In real-time data streaming, labels often represent a relationship where the input features depend on the target variable, known as a $Y \to X$ problem. This dependency can lead to a situation where the labeling may not fully align with the underlying characteristics of the data, even though it serves the application's purpose. Such a mismatch may obscure the true data structure, complicating the model's ability to accurately capture patterns. In these scenarios, internal validation measures become essential, as they provide an independent assessment of model performance without relying on potentially inconsistent labels that may not reflect the evolving geometry of the data.

For the evaluation of existing incremental clustering validation indices (iCVIs), the separation-based Partition Separation Index (PS) is selected [WY05]. This index assesses the quality of partitions and is applicable to both static and incremental clustering problems, particularly excelling at detecting under-partitioning [DSMW20], [Lug08]. Higher values of the PS index indicate better partition quality, making it a useful metric for evaluating clustering performance. However, since the PS relies on cluster centroids in its definition, methods that use centroids may achieve better results with respect to this index. An obvious limitation of this centroid-based approach is its difficulty in handling non-convex clusters and noise, which can lead to suboptimal evaluations in more complex data structures.

Incremental clustering validation indices (iCVIs), such as the selected *Partition Separation Index (PS)*, are constrained by their incremental deployment. Rather than functioning as strict validation measures, these indices serve as performance monitors, providing continuous feedback on the clustering process. This ongoing evaluation is particularly valuable in dynamic, real-time environments where clustering models must continuously adapt to evolving data distributions. However, internal validation measures specifically designed to address concept drift, such as the *Temporal Silhouette Index (TS)* [IVZ23], offer a more refined assessment. Inspired by the well-known Silhouette Index [Rou87] in static clustering, the TS index combines measures of cluster compactness and separation, incorporating temporal aspects to account for the evolving nature of the data. Higher values in the range between 0 and 1 indicate better clustering. Like the PS index, the TS index relies on cluster centroids, inheriting similar limitations when dealing with non-convex clusters or noisy data. The TS index is configured with its recommended default parameters $w = 100$, $k = 1000$, and $\varsigma = 1$.

In the context of outlier detection, we employ the ROC (receiver operating characteristic) AUC (area under curve) score as an evaluation metric because of its effectiveness in addressing imbalanced class labels. This score offers a comprehensive assessment of a model's performance by examining both the true positive rate (TPR) and the false positive rate (FPR).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The ROC curve plots the FPR on the x-axis against the TPR on the y-axis. FPR represents the probability of a false alarm, indicating how likely it is for a negative instance to be misclassified as positive. Increasing the threshold for class labeling reduces the FPR, as fewer negative instances can exceed this higher threshold. Conversely, TPR reflects the probability that a positive instance is correctly classified, and lowering the threshold can maximize this metric, though it also raises the FPR.

The ROC-AUC score quantifies the likelihood that a randomly chosen positive instance will receive a higher score for the positive class than a randomly chosen negative instance. A classifier should ideally achieve an ROC-AUC score greater than 0.5, indicating performance better than that of a random classifier [FGG+18]. This capability to evaluate performance across varying classification thresholds makes ROC-AUC particularly useful for models dealing with imbalanced class distributions, such as in outlier detection [HYS+17].

3.2.4 Experiments

Evaluation and Comparison. The evaluation consists of three sets of experiments, all adhering to the same experimental setup. Parameters related to *internal update* are configured to process 100 data points, with the initial 200 data points allocated for algorithm initialization. During the *tuning phase*, the first 10% of the dataset is used to optimize the *searched* parameters listed in Table 3.2.

1. *Tuning Phase*: Parameter optimization is performed using Optuna [CT19], selecting the best parameters from 50 combinations generated by a Tree-Structured Parzen Estimator [Wat23], with ARI as the optimization function.

2. *Analysis Phase*: Following the tuning phase, the algorithm processes the entire dataset incrementally. Performance is evaluated using ARI, PS, and TS indices, and runtime is also recorded.

The experiments are designed to align with the datasets described in Section 3.2.1:

1. *Multiple Concept Drift*: This experiments dataset features complex scenarios and diverse concept drift patterns. The goal is to assess the algorithms' ability to manage and adapt to multiple, evolving concept drifts.

2. *Specific Concept Drift*: This set of experiments focuses on how the algorithms handle distinct concept drift patterns, with and without outliers.

3. *Real-world data*: This series of tests aims to evaluate the algorithms' performance in real-world streaming data scenarios, assessing their practical applicability.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Parameter Testing. A second set of experiments examines the effects of parametrization on the performance of SDOstreamclust. In these experiments, parameters are randomly altered across 10,000 configurations using Optuna [CT19] to assess how changes impact the overall effectiveness and adaptability of SDOstreamclust. The focus is on systematically exploring different parameter combinations, as outlined in Table 3.3, which details the parameter search space, including ranges, grid sizes, and scales. The goal is to evaluate their influence on the primary performance metric, Adjusted Rand Index (ARI), in order to identify optimal parameter settings and understand the algorithm's sensitivity to various configurations. This experiment is applied to the Drifting Conglomerates dataset, designed to simulate a range of concept drift scenarios, as well as the real-world datasets from Section 3.2.1. The same setup is evaluated multiple times due to the stochastic nature of the algorithm (e.g., sampling variability), with 10 repetitions for the smaller datasets (Retail, Fertility vs Income), where the effect is more pronounced, and 3 repetitions for the larger datasets (Drifting Conglomerates, Network traffic-flows, Occupancy). Median ARI values are reported for each experiment to ensure robust results.

Table 3.3: Parameters explored in the experiments, including parameter type, value range, type of search range (linear, log, and square), and grid size. Here, an empty entry for grid size indicates a continuous grid.

	type	range	grid size	scale
k†	rate of change	[35, 800]	10	$\log_2$
x‡	rate of change	[1, 27]	9	$\log_2$
T*	temporal memory	[50, 2000]		$\log_2$
outlier_threshold	outlier thresholding	[1.1, 10]	10	sq
qv	cluster formation	[0, 0.5]		lin
zeta	cluster formation	[0.01, 0.99]		lin
chi_prop$^\diamond$	cluster formation	[0.01, 0.5]		$\log_2$

Notation: $\chi \leftarrow \lfloor \texttt{chi\_prop} \cdot \texttt{k} \rfloor$, $f \leftarrow \exp(-\texttt{T}^{-1})$, $h^o \leftarrow \texttt{outlier\_threshold}$, $\rho \leftarrow \texttt{qv}$.

†: Small datasets (e.g., Retail, Fertility vs. Income): range [25, 200], grid size 7.
‡: Small datasets (e.g., Retail, Fertility vs. Income): range [1, 19], grid size 8.
*: Retail: range [50, 250]; Fertility vs. Income: range [50, 500].
$^\diamond$: Datasets with > 3 classes (e.g., Drifting Conglomerate, Network traffic-flows): range [0.01, 0.35].

The same experiment, focused solely on outlier detection, is applied to the datasets with outlier labels in the ground truth, specifically the Drifting Conglomerates. In this context, parameters related to cluster formation, namely chi_prop and zeta, are not relevant. The parameters explored in this analysis are summarized in Table 3.4. Median ROC-AUC scores of 5 repititions are reported for each configuration.

Visualization. Our implementation provides real-time access to the current model and its active observers (a small set of data points) during the stream analysis process, enabling effective visualization of the model's performance. This study employs three well-performing configurations—small, medium, and large—based on insights from parameter testing experiments. These configurations, detailed in Table 3.5, allow for an observational visual analysis of the clustering algorithm.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 3.4: Parameters explored in the experiments related to outlier detection, including their type, value range, grid size, and scale of the search space.

	type	range	grid size	scale
`k`	model size	35 to 800	10	$\log_2$
`x`	rate of change	1 to 27	9	$\log_2$
`T`	temporal memory	250 to 2000		$\log_2$
`outlier_threshold`	outlier thresholding	1.1 to 10	20	sq
`qv`	model s./outlier thr.	0 to 0.5		lin

Notation: $f \leftarrow \exp(-\texttt{T}^{-1})$, $h^o \leftarrow$ **`outlier_threshold`**, and $\rho \leftarrow$ **`qv`**.

Table 3.5: Configurations for Different Model Sizes: Parameter settings for small, medium, and large models, illustrating the effects of `chi_prop`, `k`, and `T` on the clustering algorithm's performance in the Drifting Conglomerates dataset.

	`k`	`T`	`chi_prop`	`x`	`zeta`	`qv`
small	50	700	0.125	1	0.50	0.02
medium	200	700	0.050	2	0.45	0.10
large	800	1750	0.015	2	0.70	0.20

The analysis is conducted on the Drifting Conglomerates dataset, providing a comprehensive understanding of how the model adapts to varying parameterizations, particularly the most relevant parameters: `chi_prop`, `k`, and `T`. By comparing these different configurations, we can observe the effects of altering these key parameters, thereby gaining insights into the clustering algorithm's behavior under diverse conditions, especially in the context of concept drift.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Results and Discussion

In this chapter, we present the results obtained from the experiments, described in Section 3.2.4, and provide a discussion on their implications. A brief description of the implementation used for these experiments can be found in the Appendix I.

4.1 Evaluation and Comparison

In this section, we present the results for each set of experiments individually, followed by a summary of the overall observations. Given that some algorithms demonstrate significant sensitivity to parameter settings, the experiments involving Concept Drift and Real-world Data are conducted five times each. The results are presented as averages to account for this variability.

Specific concept drift collection. Table 4.1 presents the results of experiments conducted with this collection, which evaluates algorithm performance across a variety of specific concept drift scenarios. The table reports the median values for each metric and runtime, aggregated across all datasets within each category. The results, illustrated in Figures 4.1 and 4.2, demonstrate the superiority of both DBstream and SDOstreamclust across all settings and data types, whether stationary or exhibiting different forms of concept drift, with or without the presence of outliers. In comparison, StreamKMeans and DenStream show weaker performance, while CluStream consistently exhibits the lowest performance overall.

In *sequential* concept drift scenarios, both DBstream and SDOstreamclust experience a decline in performance. However, the drop is notably less severe for SDOstreamclust, as illustrated in Figure 4.2. This behavior can be attributed to SDOstreamclust's tendency to misclassify the initial points of emerging clusters as outliers, reflecting its cautious approach to introducing new clusters and its reluctance to prematurely split existing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Table 4.1: Performance results on the Specific Concept Drift experiments. Top results (within 5% of the best) are highlighted in bold.

		base		moving		non-stationary		sequential	
	algorithm	clean	normal	clean	normal	clean	normal	clean	normal
ARI ↑	CluSt	0.10	0.01	0.05	0.02	0.18	0.14	0.21	0.18
	DBst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.36	0.76
	DenSt	0.41	0.52	0.79	0.77	0.28	0.19	0.00	0.00
	SDOstcl	**1.00**	**0.99**	**1.00**	**0.99**	**0.99**	**0.99**	**0.97**	**0.96**
	stKMns	0.93	0.62	0.80	0.51	0.92	0.61	0.00	0.67
TS ↑	CluSt	-0.02	-0.00	0.04	0.07	0.18	-0.09	0.48	0.22
	DBst	**0.92**	**0.91**	**0.82**	**0.83**	**0.94**	**0.93**	0.32	0.65
	DenSt	0.41	0.56	0.59	0.59	0.39	0.29	-0.07	0.11
	SDOstcl	**0.88**	**0.88**	**0.81**	**0.81**	0.77	0.80	**0.54**	**0.75**
	stKMns	0.81	0.56	0.62	0.46	0.78	0.47	0.44	0.69
iPS ↑	CluSt	0.38	0.21	0.81	0.32	0.49	0.52	**0.97**	-0.43
	DBst	**1.69**	-39.46	**1.86**	-74.10	**1.56**	-30.06	0.00	-31.89
	DenSt	0.65	0.96	0.54	**1.47**	0.94	**0.99**	0.00	**0.00**
	SDOstcl	-5.15	-4.91	-6.90	-4.06	-13.61	-7.29	-17.95	-12.35
	stKMns	1.27	**1.08**	1.31	0.91	1.12	0.75	**0.97**	0.06
time (s) ↓	CluSt	89.73	128.10	106.81	142.33	24.38	20.62	104.51	180.28
	DBst	6.57	84.25	6.94	89.86	4.84	46.56	6.13	66.74
	DenSt	2.25	2.40	3.87	4.17	1.99	2.15	**1.44**	**1.50**
	SDOstcl	27.65	31.67	23.50	33.58	31.16	35.95	41.42	47.94
	stKMns	**1.78**	**1.81**	**1.68**	**1.75**	**1.55**	**1.61**	2.33	2.49

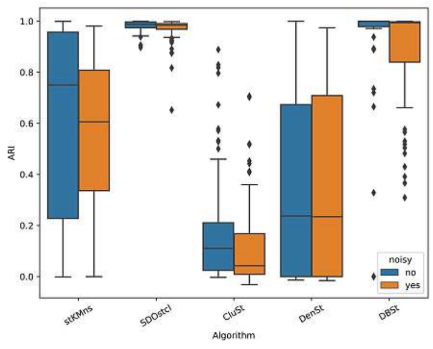
sKMns: StreamKMeans, SDOstcl: SDOstreamclust, CluSt: Clustream, DenSt: DenStream, DBst: DBstream

ones. In contrast, DBstream tends to overcluster low-density regions, fragment evolving clusters, and struggles to maintain clear separations between sequentially forming clusters. While this could also be due to a small training set in the tuning phase leading to poor parameterization, SDOstreamclust is undoubtedly far more robust with respect to the challenges posed by sequential cluster appearances.

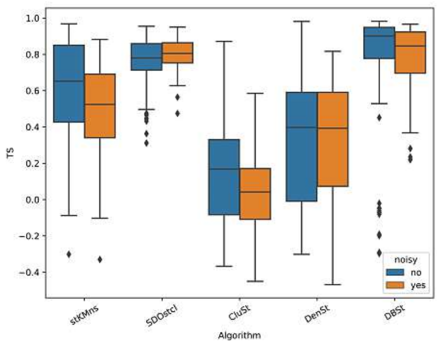
In both experiments related to concept drift, the results indicate that the PS index fails to effectively capture performance. This limitation stems from the complex, often non-convex nature of the data, which poses unique challenges in concept drift scenarios. In contrast, the TS index offers a clearer representation of performance, as illustrated in Figures 4.1 and 4.2 (compare ARI (a) with TS (b) and PS (c)).

Another important characteristic is stability, which is highlighted in both figures, showcasing the superior performance of SDOstreamclust. Its consistent results across various scenarios reflect the robustness of the algorithm, especially when compared to other methods that require extensive parameter tuning or are more sensitive to data variations. This stability aligns with the design philosophy of SDO algorithms, which utilize a minimal number of parameters and function effectively with default settings. On the other hand, CluStream's suboptimal performance can be attributed to its sensitivity to numerous parameters. Since it relies on k-means internally, which also requires careful tuning, the overall parameter adjustment for CluStream becomes inherently complex and challenging.

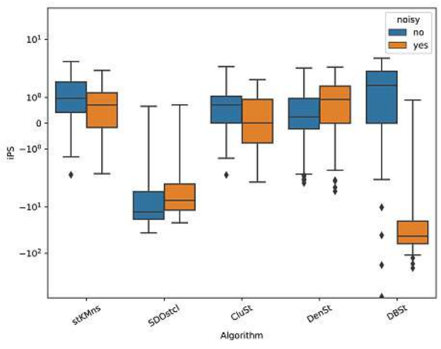
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



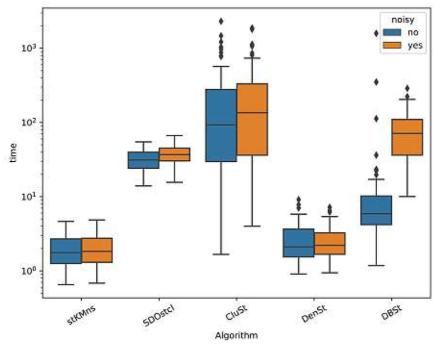
(a) ARI



(b) TS

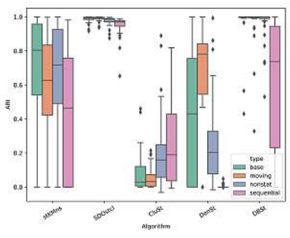


(c) PS (symlog-scale)

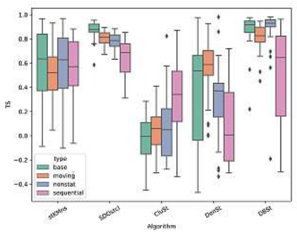


(d) runtime (log-scale)

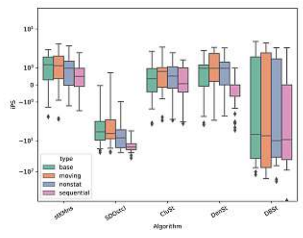
Figure 4.1: Box plots showing the aggregated performance indices from the Specific Concept Drift tests, grouped by datasets with noise (orange) and without noise (blue).



(a) ARI



(b) TS



(c) PS (symlog-scale)

Figure 4.2: Box plots showing the aggregated performance indices from the Specific Concept Drift tests, grouped by dataset types: base, moving, non-stationary, and sequential.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

When considering runtime performance, StreamKMeans and DenStream emerge as the fastest algorithms. Following closely are SDOstreamclust and DBStream, with SDOstreamclust demonstrating remarkable stability in its runtime efficiency. Notably, DBStream requires significantly more computational effort when handling noisy data, which can negatively impact its overall performance. In contrast, CluStream exhibits suboptimal runtime characteristics. However, it is important to note that SDOstreamclust can scale efficiently by leveraging batch processing (in experiment size 1) and input buffering (in experiment size 10) to optimize runtime, making it a robust choice for various data scenarios.

Drifting conglomerates and real-world data. The results of these tests are summarized in Table 4.2. Again, due to the sensitivity of some algorithms' parameters, we show the median results of five repetitions.

Table 4.2: Results of the Drifting conglomerates and Real-world data experiments. Top results (within 5% of the best) are highlighted in bold.

	algorithm	Drifting Conglomerates	Fertility vs Income	Network traffic-flows	Occupancy	Retail
	CluSt	0.088	0.001	0.265	0.010	0.113
ARI ↑	DBst	0.345	**0.524**	0.000	**0.340**	**0.939**
	DenSt	0.193	0.001	0.001	0.000	-0.000
	SDOstcl	**0.953**	0.422	**0.461**	0.118	0.705
	stKMns	0.396	0.009	0.060	0.006	0.776
	CluSt	0.231	0.141	0.406	**0.816**	0.312
TS ↑	DBst	0.536	**0.537**	0.585	0.767	**0.677**
	DenSt	0.577	0.057	**0.694**	-0.054	0.279
	SDOstcl	**0.783**	0.413	0.326	0.647	0.529
	stKMns	0.604	0.258	0.595	**0.821**	**0.698**
	CluSt	-1.375	**1.011**	0.455	**0.976**	0.384
iPS ↑	DBst	-18.522	-0.571	**0.894**	-2.441	1.195
	DenSt	**0.757**	0.663	-1.719	0.000	0.975
	SDOstcl	-13.257	-4.437	-41.212	-34.916	-7.904
	stKMns	-1.777	1.446	0.196	0.970	**1.350**
	CluSt	8.422	3.550	1154.075	6.612	0.244
time (s) ↓	DBst	9.085	0.875	9.262	3.368	0.160
	DenSt	**1.717**	0.370	**1.720**	0.858	0.091
	SDOstcl	20.558	0.982	32.950	25.220	0.217
	stKMns	1.082	**0.260**	2.761	**0.604**	**0.064**

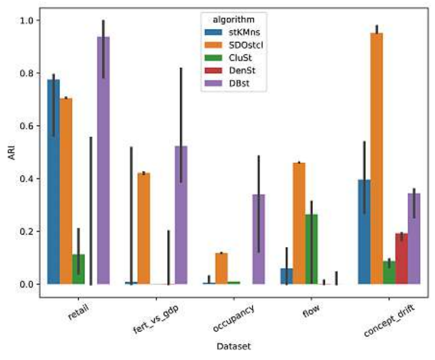
sKMns: StreamKMeans, SDOstcl: SDOstreamclust, CluSt: Clustream, DenSt: DenStream, DBst: DBstream

Among the algorithms tested, SDOstreamclust emerges as the most effective for the Conglomerative Drifting dataset, demonstrating a substantial advantage over the others. It uniquely manages to handle the simultaneous occurrence of various types of concept drift specific to this dataset.

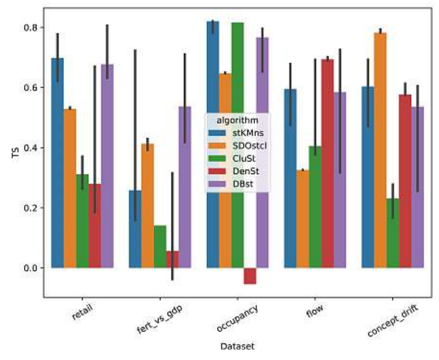
For the Retail dataset, DBstream achieves the best clustering performance. However, both StreamKMeans and SDOstreamclust occasionally merge the upper clusters to some extent. This behavior is likely due to the proximity of the clusters, their simultaneous

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

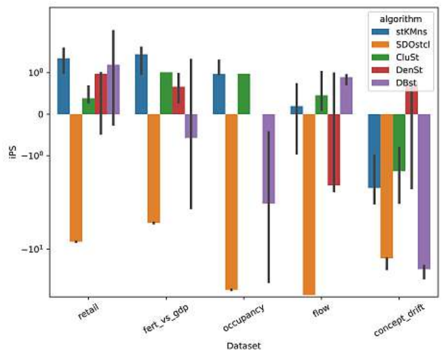
evolution, and the small number of data points in the intermediate cluster, which may be considered local outliers near the upper cluster (Fig. 3.7.d).



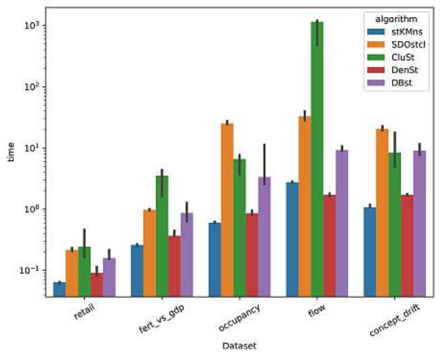
(a) ARI



(b) TS



(c) PS (symlog-scale)



(d) runtime (log-scale)

Figure 4.3: Bar plots for the Real-world and Drifting Conglomerate performance results.

In the Fertility vs. Income dataset (Fig. 3.7.c), both DBstream and SDOstreamclust are able to differentiate the two evolving trends (European and non-European countries). However, SDOstreamclust experiences a decline in performance initially, likely due to the close proximity of the two groups, which causes overclustering. This might be because the model is still in its formative stages and not yet fully developed.

The relatively small size of both the Retail and Fertility vs. Income datasets allows the startup phase to have a more pronounced impact on the overall performance of all algorithms, as they have less time, relative to the dataset size, to stabilize and form a robust representation. The limited dataset size may require a more specific setup, such as fine-tuning algorithm parameters, to ensure adequate model stability and representation throughout the experimental runs.

The Occupancy dataset (Fig. 3.7.b) poses unique challenges. Only DBstream and SDOstreamclust manage to track the data's dynamics, though their results do not match

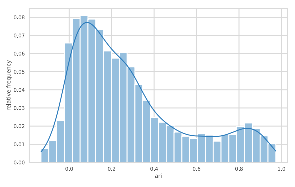
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the Ground Truth. The difficulty arises from the slow, continuous changes in most features (temperature, humidity, CO_2), leading to overlapping Ground Truth classes. The feature light does change abruptly, but this signal is obscured by the gradual drifts in other features. When a new light period emerges or ends, DBstream and SDOstreamclust tend to create new clusters rather than linking to previous ones. CluStream and StreamKMeans achieve high internal evaluations (PS and TS) because they almost always detect a single cluster throughout the stream, preventing them from being completely zeroed out as DenStream is, since internal validation struggles with single-label clusterings.

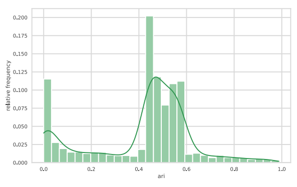
The complexity of the Network-traffic dataset (Fig. 3.7.a) arises from its sequential class transitions and the considerable variability differences among its features. SDOstreamclust is the only algorithm that effectively captures this dynamic and distinguishes the key classes. However, it frequently detects subclusters and outliers within the primary classes, attributed to significant intra-class density variations.

4.2 Parameter Testing

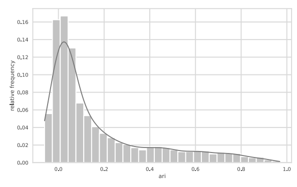
In the experiments described in the previous Section 4.1, we evaluated our algorithm with a primarily out-of-the-box configuration, making only minimal tuning to the temporal memory parameter T and the outlier threshold parameter h^o. Despite this limitation, the algorithm demonstrated competitiveness with existing alternatives. In this section, we go deeper into the effects of parameterization and its interaction with individual characteristics of the data.



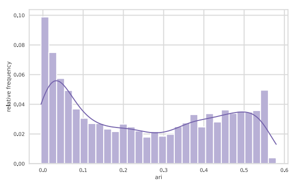
(a) Drifting Conglomorate



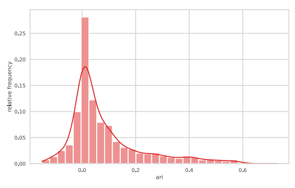
(b) Retail



(c) Fertility vs Income



(d) Network traffic-flows



(e) Occupancy

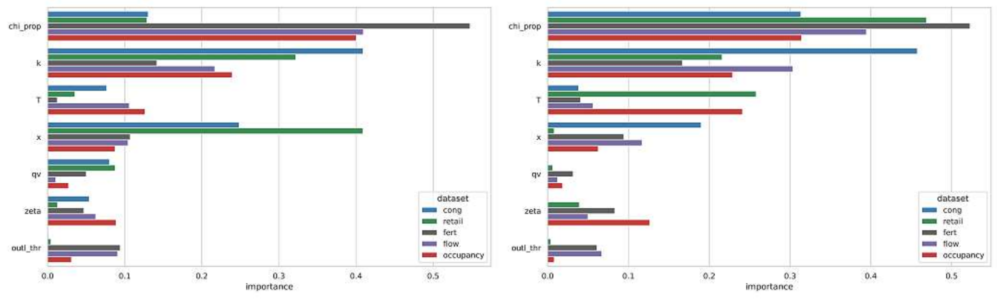
Figure 4.4: Histograms of Adjusted Rand Index (ARI) scores for all experiments, representing the performance across different datasets. The x-axis shows the ARI scores, while the y-axis indicates the frequency of runs that achieved each score.

To achieve this, we conduct a comprehensive evaluation of various parameters, as detailed in Table 3.3, across the entire dataset, reporting the Adjusted Rand Index (ARI). We also

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

investigate unconventional parameter setups that may yield suboptimal scores, which, while seemingly counterintuitive, provide valuable insights into the algorithm's behavior. Notably, we exclude runs with a replacement rate exceeding 0.5, as our streaming setup prioritizes lightweight processing and speed while maintaining accuracy; thus, we do not aim for solutions that fully process all points. This filtering results in approximately half of the evaluated setups. The overall distribution of results is presented in Fig. 4.4. From this point forward, the different datasets will be consistently color-coded in the figures, as shown in this histogram illustration. The illustrations reveal that Drifting Conglomerate (a) and Network Traffic-Flows (d) require less specific setups to achieve optimal results, while the other three datasets (Retail, Fertility vs Income, and Occupancy) benefit more from careful fine-tuning.

Parameter importance and correlation. We analyzed the importance of each parameter using a Lasso regression model to assess their contributions to overall performance, as illustrated in Fig. 4.5. This analysis encompasses all applied setups and focuses specifically on the top 50%. The rationale behind this approach is that excluding poor combinations is often straightforward without requiring extensive expert knowledge. Thus, our focus is on analyzing the interactions between the parameters of SDOstreamclust in decent or good setups, rather than solely identifying non-viable configurations.



(a) All runs included (b) Only Top 50% results

Figure 4.5: Parameter importance analysis for all parameters and datasets, contrasting overall results with those from the top-performing half of the experiments.

When comparing both figures, the x parameter is most significant in the Retail dataset. This is due to the effectiveness of setups with small models (i.e., small k) for this dataset. However, if x (which controls the number of observers observing a processed point) is set too high, the model struggles to distinguish effectively between good and poor representatives of the underlying data (i.e., observers). In Figure 4.5 (b), it is clear that this parameter becomes unimportant when chosen appropriately, but can lead to suboptimal outcomes when selected poorly. Similarly, the qv parameter loses its importance for the better-performing setups in the synthetically generated Drifting Conglomerate dataset. This parameter, denoted as ρ in our algorithm, adds robustness against noise, outliers, and uncertainty in the data, albeit at the cost of reducing the model's capacity to capture patterns. Although this dataset contains artificial noise

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

and outliers, it is likely easier to distinguish patterns compared to those from real-world data, which often have many underlying factors not covered by the data, possibly even unknown, that contribute to uncertainty. Given the presence of various clusters and patterns, a larger model size is advantageous for this dataset. Thus, selecting `qv` becomes straightforward, focusing primarily on retaining a large enough model while requiring only minimal safeguards against noise and outliers.

Overall, the most important parameters identified are `chi_prop`, which controls cluster formation by determining the likelihood of observers connecting with one another, and `k`, the most relevant parameter for defining the model size, representing the number of observers. The algorithm calculates χ as $\chi \leftarrow \lfloor \texttt{chi\_prop} \cdot \texttt{k} \rfloor$. Other parameters, such as `T` and `x`, appear to be significant only for specific dataset characteristics. For instance, the Retail dataset, as shown in Fig. 3.7 (d), exhibits sudden changes in trend at various points, indicating that `T` may need to be tailored accordingly. Similarly, the Occupancy dataset also experiences abrupt changes. It is crucial to note that such specific fine-tuning may not be feasible, as these changes cannot be anticipated without comprehensive prior knowledge of the data in a real-world context. In contrast, the synthetic Drifting Conglomerate dataset features a greater number of classes or clusters, suggesting that the observer parameter `x` may need to be more specific; thus, a lower `x` relative to the model size is potentially important in this scenario. In general, a sufficiently low and consequently specific `x` provides robustness against noise. It should only be increased if there is insufficient incoming information (processed data points) relative to the model size; otherwise, it may be more beneficial to reduce the model size.

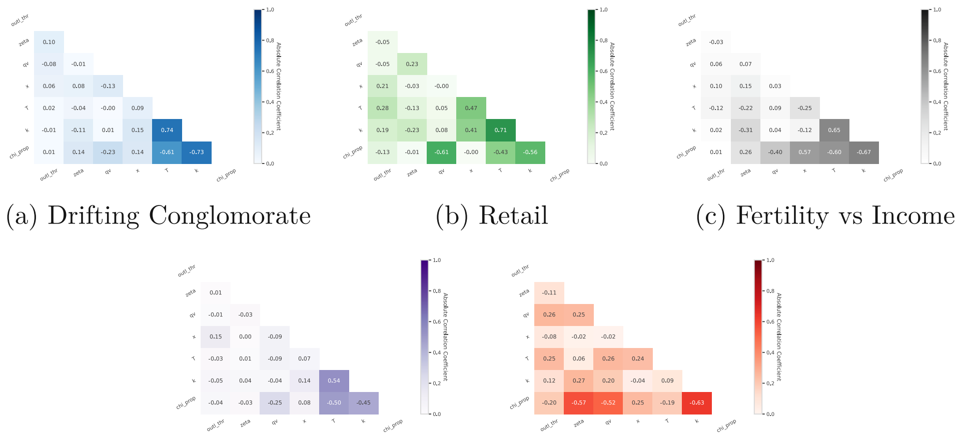
The other parameters, `qv`, `zeta`, and `outlier_threshold`, are of minor importance for these datasets and appear less relevant. Notably, `outlier_threshold` governs a slightly different application related to outlier detection and is tested later separately

As mentioned, the parameters interact with each other. To further investigate this, we examined correlations among parameters, focusing exclusively on setups that achieved at least 80% of the highest Adjusted Rand Index (ARI) score. This approach aids in identifying which parameters exhibit strong interactions for successful configurations, as illustrated in Fig. 4.6, where the correlations are represented as heatmaps.

Not surprisingly, in all plots, we observe significant interactions between the key parameter `chi_prop` and both `k` and `T`, as well as with `qv` and `zeta`, especially in the Retail and Occupancy datasets. Notably, `chi_prop` strongly depends on the model size, largely driven by `k`. A finer granularity of observers allows for more precise tuning of their connectivity, while a denser model provides richer representations of the underlying data. The combination of `T` and `k` controls the replacement rate and the model's fading memory effect, further influencing the role of `chi_prop`. This interaction, and its correlation with other key parameters, is illustrated in Fig. 4.7.

The parameter `qv` refines the model in the Retail dataset by pruning observers that are insufficiently connected to clusters (as seen in the sparse upper section of the brown cluster in Fig. 3.7 (d)). This results in a more specific model, which in turn demands

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



(a) Drifting Conglomerate (b) Retail (c) Fertility vs Income

(d) Network traffic-flows (e) Occupancy

Figure 4.6: Illustration of correlations among the parameters of SDOstreamclust across all datasets, highlighting the top results that fall within 25% of the best evaluation scores, measured by the Adjusted Rand Index (ARI).

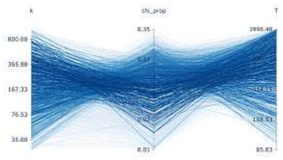
a lower value of chi_prop. A similar effect may be observed in the Occupancy and Fertility vs. Income datasets. Furthermore, the interaction with zeta stems from the noisy spatial and temporal patterns in the Occupancy dataset, not only on a global scale but also within the cluster shape and geometry, which necessitate global thresholds. These thresholds reduce model granularity, requiring larger values of chi_prop to prevent cluster fragmentation.

The parameter x interacts with k and T, as discussed at the beginning of this section. The more information available, the more specific the model can be trained (e.g., a lower x). This interaction is visible to some degree across all analyzed datasets, particularly for those that perform well with smaller models. The unusually high correlation with chi_prop suggests that x determines the specificity and granularity of the model more directly, while the effects of model size and temporal behavior are comparatively less significant.

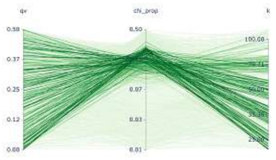
The outlier_threshold parameter is particularly relevant in datasets that appear noisy but lack an explicit outlier class label. Thus, an optimal setup aims to avoid declaring outliers unnecessarily. This is crucial because it is generally unsafe to assume that outliers are absent. Experiments for this parameter specifically follow below.

Replacement rate. Lastly, both k and T interact strongly across all datasets, except for the Occupancy dataset, as they jointly determine the replacement rate, $\frac{k}{T}$, which significantly impacts the model's adaptability. However, this also increases the algorithmic cost, as maintaining the graph via an M-tree and performing node (observer) addition/removal operations—costly actions with a time complexity of $O(\log n)$ for an M-tree of size n—are directly influenced by the replacement rate. Thus, both the replacement rate and

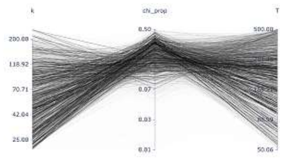
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



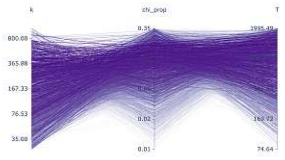
(a) Drifting Conglomerate



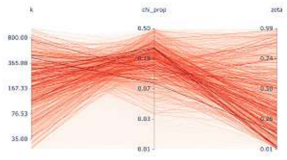
(b) Retail



(c) Fertility vs Income



(d) Network traffic-flows



(e) Occupancy

Figure 4.7: Illustrating the interaction between parameter `chi_prop` and its most correlated parameters using parallel coordinates plots across all datasets. Good setups are highlighted with darker coloring.

the number of observers are key factors in determining the overall algorithmic cost and runtime. The best results, filtered by the number of observers `k` and the threshold on the replacement rate, are shown in Table 4.3.

It is evident that as more information is processed (i.e., with a higher replacement rate), the results improve; however, this increase comes with higher algorithmic costs. Second, as observed in the Retail and Fertility vs. Income results, a larger model size (more observers) is beneficial only if it receives sufficient information to remain updated in the dynamic environment of data streaming analysis. If new data arrives too slowly (i.e., low replacement rate), the model becomes overly static, leading to a drastic deterioration in performance. In smaller datasets with only 2 or 3 classes (such as Retail and Fertility vs. Income), smaller model sizes prove to be more suitable due to the limited incoming data, which makes it challenging to maintain larger models dynamically as required.

Another observation is that the replacement rate is significantly more important for datasets featuring incremental drift, where clusters move spatially over time (see Fig. 3.7 (c) and (d), Fig. 3.5, and Fig. 4.1 (d)). In contrast, the performance of the Network Traffic Flows dataset is much less influenced by the replacement rate due to its characteristic of sequentially appearing clusters (see Fig. 3.7 (a) and Fig. 4.1 (b)). A similar, albeit slightly lesser, importance can be observed in the Occupancy dataset, which exhibits sequential but recurring characteristics (see Fig. 3.7 (b)). This indicates that a well-tuned replacement rate is crucial for effectively addressing incremental drift.

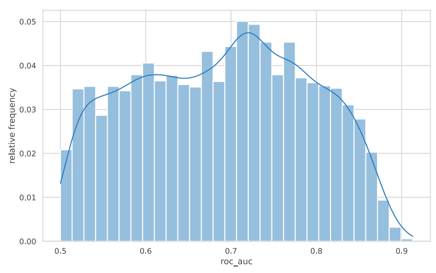
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Table 4.3: Best results (ARI) across all datasets, filtered by configuration with respect to k (number of observers) and the replacement rate threshold. Lighter text indicates results inferred from a smaller threshold, while top results (within 3% of the best) are marked in bold.

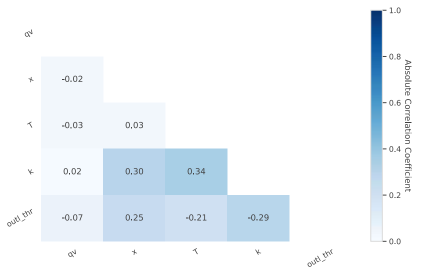
		replacement rate				
	k	≤ 0.10	≤ 0.15	≤ 0.25	≤ 0.35	≤ 0.50
Conglomerate Drift	35	0.754	0.817	0.817	0.817	0.817
	50	0.844	0.844	0.844	0.844	0.844
	70	0.844	0.863	0.873	0.873	0.873
	100	**0.867**	**0.912**	0.912	0.912	0.912
	140	**0.876**	**0.906**	**0.939**	**0.939**	0.939
	200		**0.896**	**0.940**	**0.963**	**0.963**
	280		0.849	**0.960**	**0.961**	**0.970**
	400			0.919	**0.943**	**0.973**
	560				**0.954**	**0.959**
	800					0.934
Retail	25		**0.760**	**0.900**	**0.960**	**0.988**
	35		0.587	0.866	**0.960**	**0.970**
	50			0.733	0.908	0.955
	70				0.769	0.938
	100					0.795
Fertility vs Income	25	**0.806**	**0.926**	**0.969**	**0.969**	**0.969**
	35	0.723	**0.910**	0.910	**0.942**	**0.942**
	50		0.781	0.926	0.926	**0.962**
	70		0.759	0.853	0.934	0.934
	100			0.781	0.897	**0.942**
	140				0.797	0.897
	200					0.815
Network traffic-flows	35	0.536	0.536	0.538	0.538	0.538
	50	**0.556**	**0.556**	0.556	0.556	0.556
	70	**0.548**	**0.558**	0.558	0.558	0.558
	100	**0.553**	0.553	0.557	0.558	0.558
	140	**0.543**	**0.557**	0.557	0.558	0.558
	200		**0.572**	**0.572**	**0.572**	**0.572**
	280		0.527	**0.572**	**0.572**	**0.572**
	400			**0.580**	**0.580**	**0.580**
	560				**0.560**	**0.578**
	800					**0.566**
Occupancy	35	0.526	0.526	0.526	0.526	0.526
	50	0.560	0.560	0.560	0.560	0.560
	70	**0.596**	0.596	0.596	0.596	0.596
	100	**0.591**	0.591	0.591	0.591	0.591
	140	0.571	0.603	0.635	0.635	0.635
	200		**0.630**	**0.728**	**0.728**	**0.728**
	280		0.362	**0.713**	**0.713**	**0.713**
	400			0.628	0.628	0.628
	560				**0.722**	**0.722**
	800					0.476

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

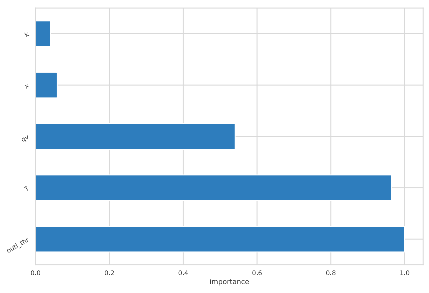
Outlier detection. In Fig. 4.8, similar illustrations as before are employed to present the evaluation results for outlier detection on the Drifting Conglomerates dataset. The histogram indicates overall stable performance across all setups. Not surprisingly, the parameter `outlier_threshold` emerges as a crucial factor for this application of data stream analysis. However, when focusing on the top 25% of setups, the temporal memory parameter `T` proves to be even more significant, as it is essential for the model to adapt effectively to changes in the data.



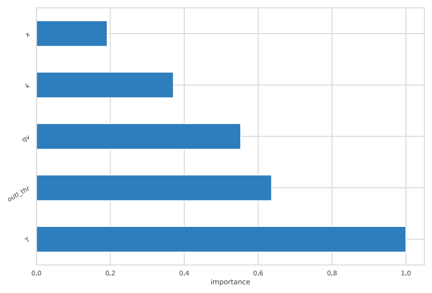
(a) Histogram



(b) Correlation (score > 80% of best)



(c) Importance (all)



(d) Importance (top 25%)

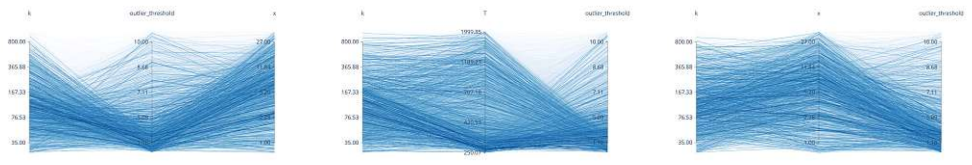
Figure 4.8: Visualization of outlier detection results on the Drifting Conglomerates dataset

In contrast to data clustering, the specificity of the model is less critical for outlier detection; thus, setups with larger `x` values are permissible and potentially advantageous to some extent. As illustrated in Fig. 4.9 (c), a more generalized model allows for smaller outlier thresholds. This is because the model (set of observers) primarily needs to cover normal points, enabling generalization without requiring specificity in cluster shapes. Similarly, the number of observers, `k`, which determines the model size, is less critical in outlier detection compared to its significance in clustering applications. The parameter `qv` serves a similar purpose as in data clustering, focusing on noise cleansing.

Fig. 4.9 (b) demonstrates that smaller values of `T` are advantageous, indicating that the model must be able to adapt to the dynamic dataset. Additionally, we observe that larger values of `T` strongly necessitate a low `outlier_threshold`. This is likely

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

because the model is too slow to adapt in such setups and compensates by setting `outlier_threshold` to a very low value, requiring that outdated observers be hit with precision to misclassify an outlier as a regular class. Despite this specificity, effective setups remain possible, not only with high `x` values (which yield a highly generalized model detrimental to clustering applications) that often accompany higher values of `k` and `T` (due to the need for a replacement rate lower than 0.5). Dynamic setups (lower `T`) do not necessitate an overly generalized approach via high `x`.



(a) `outlier_threshold` (b) `T` (c) `x`

Figure 4.9: Visualization of outlier detection results on the Drifting Conglomerates dataset: (a) Parallel coordinates plot for `outlier_threshold`, (b) for `T`, and (c) for `x`. Darker colors indicate better performance.

It is important to note that the parameters `T` and `k` show less correlation compared to their roles in clustering applications. This indicates that the replacement rate is not as critical for outlier detection in the analyzed dataset. This point is further supported by Table 3.4, which demonstrates that better-performing setups can have a higher replacement rate, but to a lesser degree. The table also confirms that outlier detection performs well across a variety of setups, as `k` influences the ranges of `T` and `x` as well.

Table 4.4: Best AUC-ROC results for the Drifting Conglomerates dataset, filtered by configurations regarding `k` (number of observers) and various thresholds for the replacement rate. Results inferred from smaller thresholds are presented in lighter text, while the top results (within 2% of the best score) are highlighted in bold.

	replacement rate				
k	≤ 0.10	≤ 0.15	≤ 0.25	≤ 0.35	≤ 0.50
35	0.853	**0.878**	0.878	0.878	0.878
50	**0.876**	**0.880**	**0.899**	**0.899**	**0.899**
70	**0.869**	**0.871**	**0.887**	**0.893**	0.893
100	**0.867**	**0.886**	0.886	**0.891**	**0.913**
140	**0.866**	**0.876**	**0.887**	**0.887**	**0.907**
200		**0.876**	0.876	**0.882**	0.886
280		0.803	0.869	0.881	0.892
400			0.848	0.878	0.878
560				0.855	0.879
800					0.871

It is important to note that this dataset is synthetic, which comes with the significant drawback that the data shape, aside from the noise, is clearly determined by the available

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

features. Consequently, there are no uncertainties or unknown factors or dimensions that could disturb the algorithm, both in the feature space and in the target space.

4.3 Visualization

In this section, we present the results of our observational visual analysis conducted on the Drifting Conglomerates dataset using the configurations outlined in Table 3.5. The visualizations illustrate how the different model sizes (small, medium, and large) and their respective parameter settings influence the clustering algorithm's performance in the presence of concept drift.

First, we look at the medium configuration, concentrating on the key parameters `chi_prop` and `T`. The visualization adopts a film-like approach, presenting snapshots of the current model (its observers) at regular intervals to illustrate how it evolves with incoming data. Figure 4.10 showcases the effects of varying `chi_prop`, while Figure 4.11 illustrates the impact of altering `T`. Each snapshot displays the ground truth data on the left, representing a time window around the moment the model snapshot is captured, with the current active model presented on the right.

Focusing on the snapshots from Figure 4.10 (b) as a form of ground truth, we can compare it to its neighboring configurations with adjusted `chi_prop`. On the left, where `chi_prop` is low, we observe that the spiraling cluster tends to fragment compared to the outer ground truth. This fragmentation occurs because older observers are gradually replaced, leading to sparser data at those positions in the moving cluster, which the model struggles to accommodate. Consequently, the model often misinterprets the new incoming denser data as a new cluster, causing the old fading data—containing vital cluster history—to become fragmented and lost.

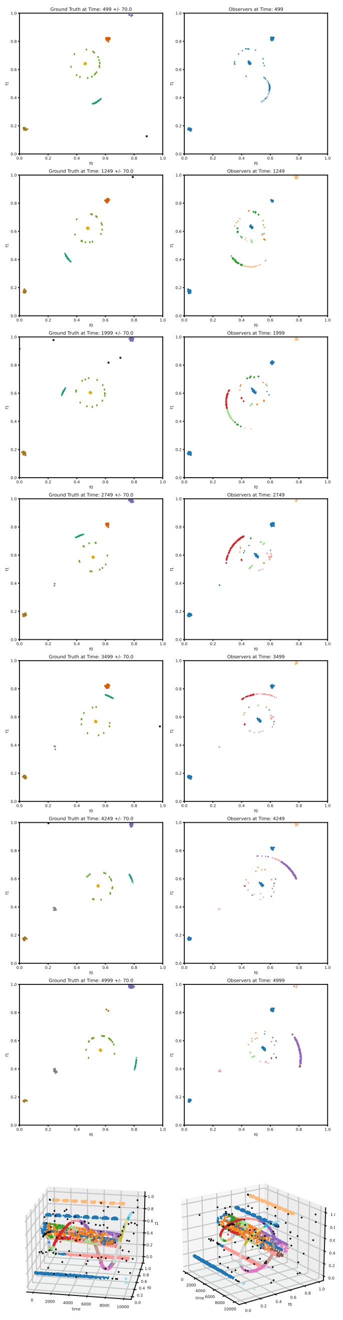
Conversely, on the right, with a higher `chi_prop`, the two challenging clusters (the sparse circle and the pointy cluster in the center) falsely connect regularly. When another moving cluster (the spiral) approaches, the clusters merge and are perceived as one (for a short time even the in the ground truth brown colored recurring class joins) for some time by the method in this configuration.

Generally, the sparse circle in combination with the pointy cluster in the center appears in the good medium model-sized setup as the most challenging. Constantly maintaining separation while it moves, combined with the additional difficulty of sparsity, poses a significant challenge. The preferred solution seems to involve allowing fragmentation of the sparse circle in favor of remaining clearly separated from the pointy central cluster.

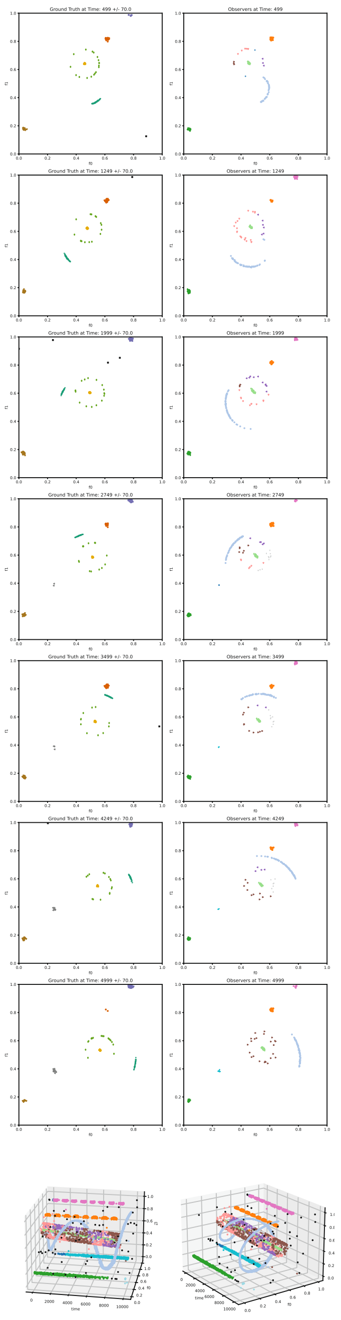
Similarly, the parameter `T` is altered, again using the configuration with medium model size. Here, we can clearly see how this parameter affects the speed at which older data is replaced. This is particularly evident when observing the length of the tail of the spiraling cluster observers and the lengthy or more pointy appearance of the moving pointy cluster at the center of the sparse circles, as illustrated in Figure 4.11.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

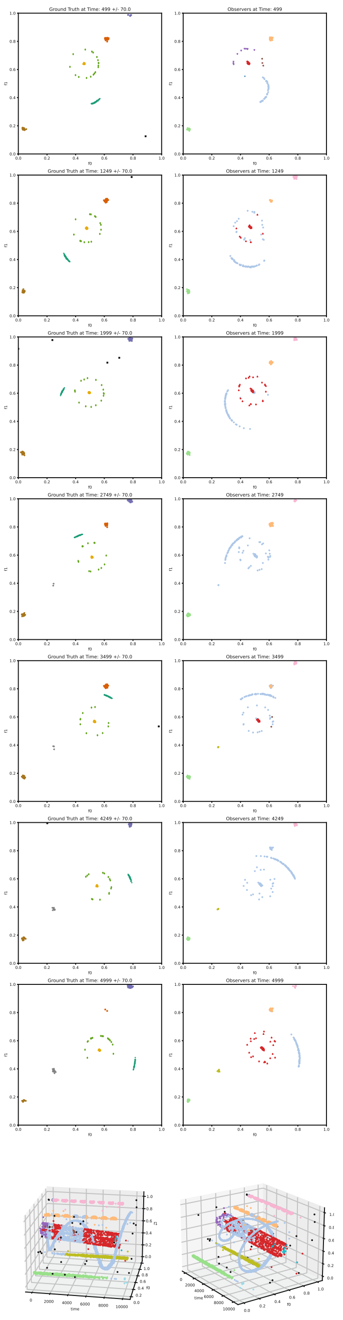
Figure 4.10: Visualization of the evolution of the model observers with varying `chi_prop` values across different frames. Each column presents a sequence of snapshots taken at specific time intervals, illustrating how the model adapts to incoming data. On the left side of each column, the frames depict the current state of the model, while the final frame in each column shows the predicted model state. At the end of the sequence, the overall results are visualized in a 3D plot, where its two features are plotted against time.



(a) `chi_prop` $= 0.02$



(b) `chi_prop` $= 0.05$



(c) `chi_prop` $= 0.125$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

In the setup with a smaller T, the plot of the overall prediction reveals that the model struggles to remember recurring clusters, treating each appearance as a novelty, and fails to effectively manage the sparse circle, as no robust history can be maintained. The memory fades too quickly, and the replacement rate is insufficient to compensate with new, well-representing observers.

While the effect on classification is not as severe for higher T upon visual inspection, if the model makes an incorrect initial classification, it becomes rigid and difficult to correct, leading to worse performance than in more optimal configurations.

The last comparison examines three well-performing configurations presented in Table 3.5, evaluating their different appearances and abilities to solve the problem at hand. It's important to note that the smallest configuration performs significantly worse than the larger models (small: 0.86, medium: 0.97, large: 0.90 ARI), but it is still impressive how such a small model is able to capture 7 distinct classes, which were designed to be challenging with respect to concept drift. The main issue the small model faces is handling sparse data, while other challenges, including recurring and moving clusters, appear to be manageable.

The larger model also struggles with the sparse circle cluster because it is too rigid; once it forms an incorrect representation (often fragmented into two or more clusters), it has difficulty correcting this. The reason for this is that a larger k also requires a larger T, which makes the model more robust to some extent but also more rigid (the larger the model, the more information is necessary to force change). It's worth noting that as the model size increases, the algorithmic cost rises, which could be a significant factor in practical applications.

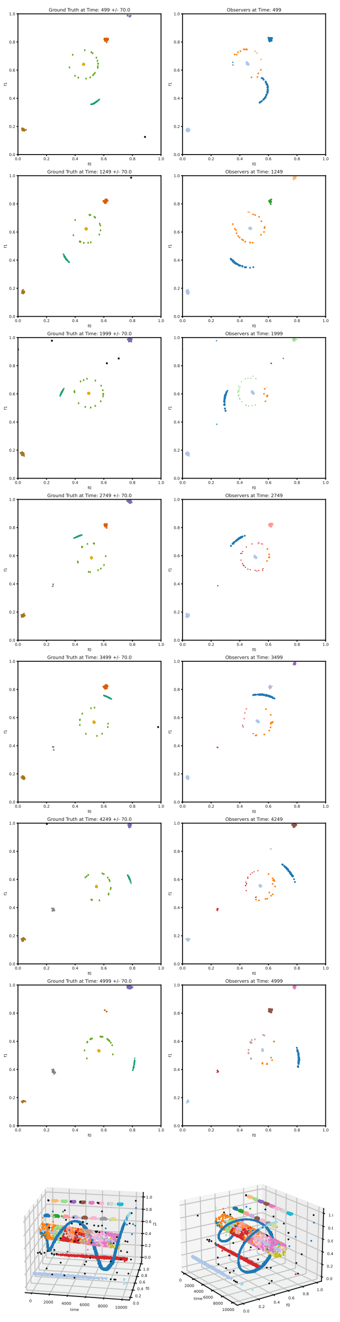
Finally, in the first snapshots of all models, but especially in the larger models, we observe initial misclassification during the warm-up phase (which is set to a relatively high value in this experiment, not making it difficult for the algorithm), from which the algorithm needs to recover as the model fully grows. As demonstrated by our small model, having a reduced model size alone does not necessarily mean that data is poorly represented. This suggests a potential issue with parametrization during the warm-up phase. It may be worthwhile to explore adapting χ, and potentially other parameters, according to the actual current model size during this phase to mitigate these initial errors.

4.4 Key Findings

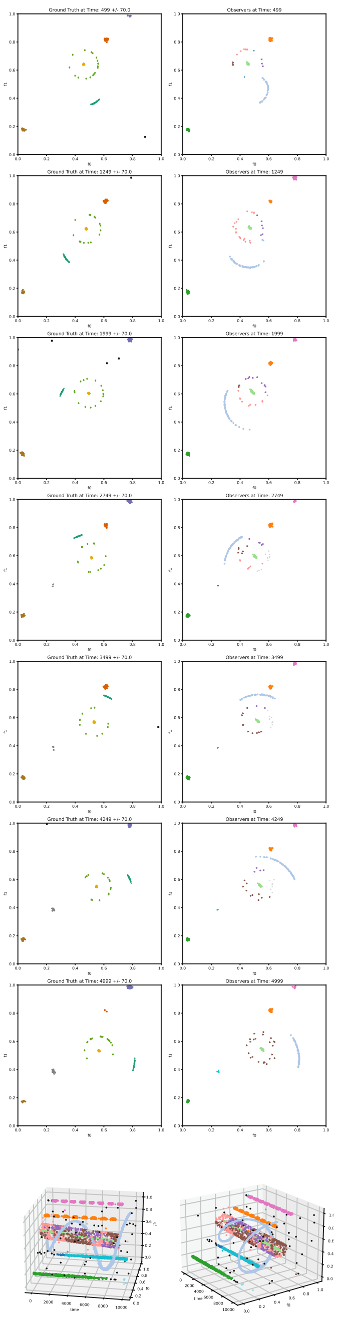
The proposed algorithm is evaluated in an out-of-the-box setup where only two parameters, T and `outlier_threshold`, are tuned to demonstrate its robustness in parameterization. It is systematically compared against competitor algorithms, showcasing effectiveness and competitiveness across all datasets, thus supporting the claims made. The evaluation includes both real-world and synthetic datasets specifically designed to simulate various forms of concept drift, representing evolving data of different types.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

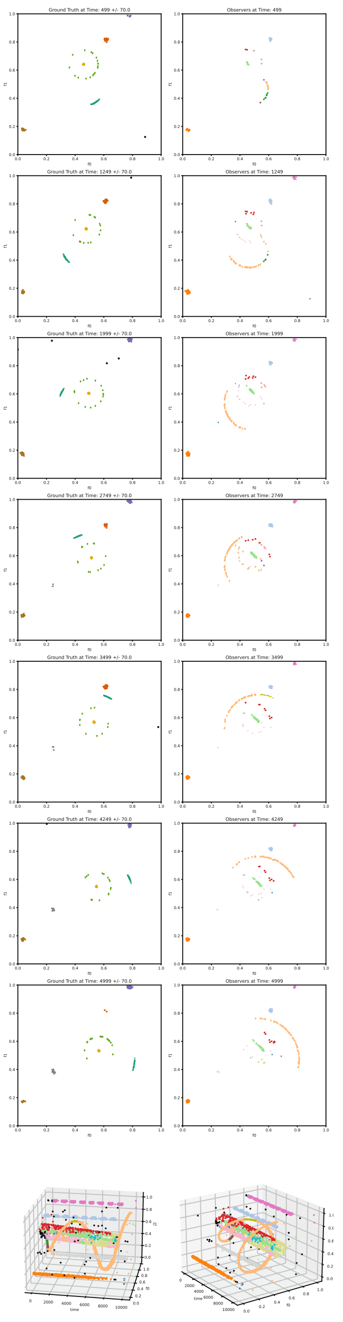
Figure 4.11: Visualization of the evolution of model observers with varying `T` values across different frames. Each column presents a sequence of snapshots taken at specific time intervals, illustrating how the parameter `T` affects the model's ability to retain and adapt to incoming data. On the left side of each column, the frames depict the current state of the model, while the final frame in each column shows the predicted model state. At the end of the sequence, the overall results are visualized in a 3D plot, highlighting the interaction of the model with the evolving data stream.



(a) `T` = 350



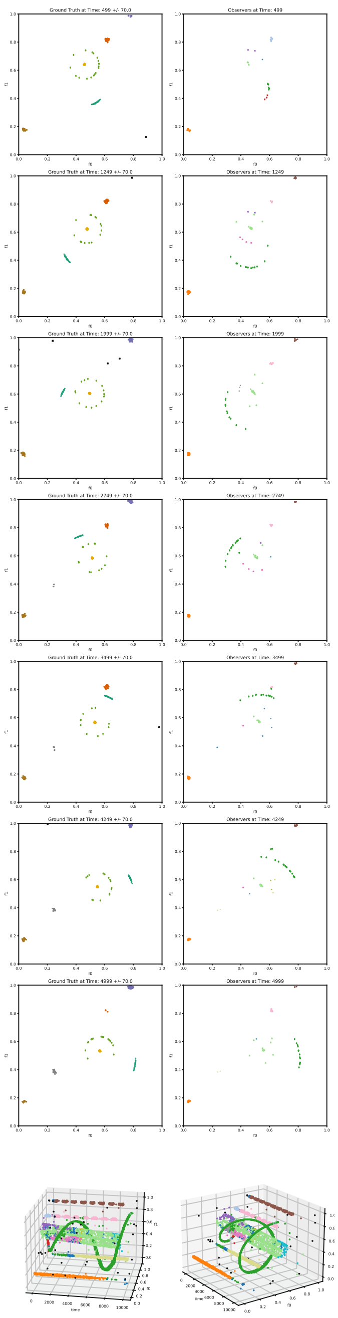
(b) `T` = 700

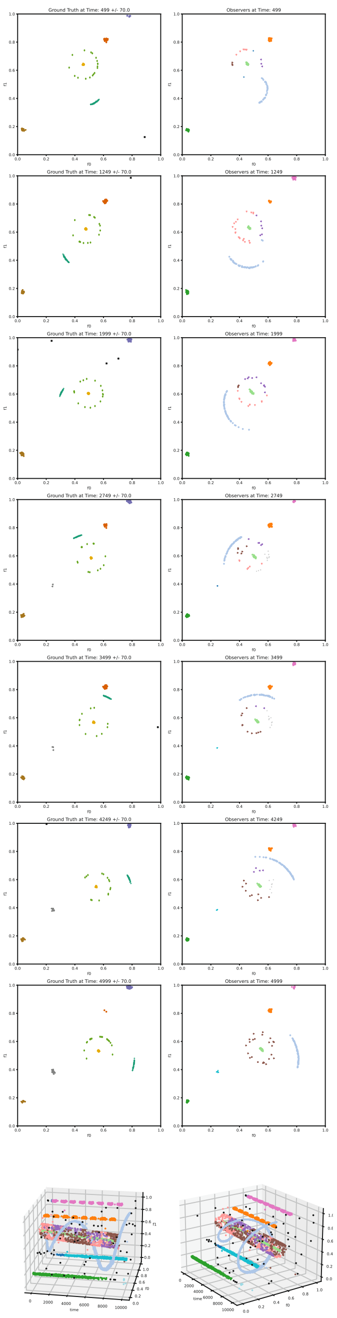


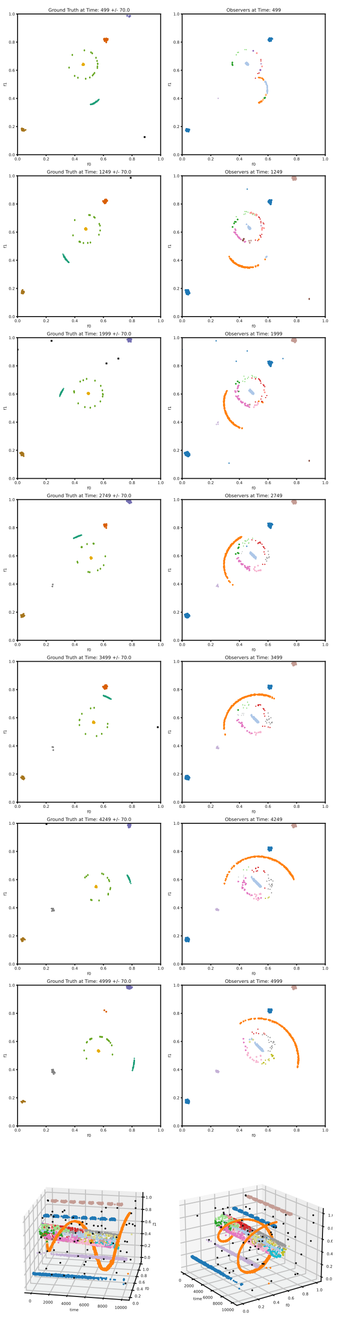
(c) `T` = 1400

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Figure 4.12: Visualization of the evolution of model observers with varying model size k across different frames, focusing on well-performing configurations, see Table 3.5. Each column presents a sequence of snapshots taken at specific time intervals, illustrating how different model sizes influence the model's ability to retain and adapt to incoming data. On the left side of each column, the frames depict the current state of the model, while the final frame in each column shows the predicted model state. At the end of the sequence, the overall results are visualized in a 3D plot, highlighting the interaction of the model with the evolving data stream.


(a) $k = 50$


(b) $k = 200$


(c) $k = 800$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Parameter testing identifies three critical parameters: `chi_prop`, which governs cluster formation and observer connectivity; `k`, which represents the number of observers and defines the model size; and `T`, which establishes a time window, influencing both temporal memory and the replacement rate in conjunction with `k`. While other parameters help fine-tune the algorithm to specific data characteristics and are relatively intuitive to adjust, these three parameters are essential for overall performance. Despite variations in parameter settings, the algorithm exhibits robustness across a wide range of configurations, as demonstrated in Table 4.3. This adaptability underscores its versatility in diverse environments.

Promising improvements regarding the implementation's parametrization that may be worth considering include providing users the option to select the replacement rate as an independent parameter, either linked to or decoupled from the relationship between `T` and `k` ($\texttt{replacement\_rate} = \frac{\texttt{k}}{\texttt{T}}$). Additionally, since parameter `x` interacts with the model size, it may also be beneficial to explore the possibility of defining it as a relative value to the number of observers `k`.

A question arises as to whether embedded outlier detection with SDOstreamclust operates effectively under similar parametrization as clustering. The key distinction is that, in outlier detection, the model must be more generalized—a difference that is unsurprising when considering the two applications. Clustering aims to differentiate the shapes of clusters, whereas outlier detection only requires a characterization of normal data. The main parameter influencing generalization is `x`, but as observed, SDOstreamclust demonstrates parameter robustness with respect to `x` even in clustering tasks.

The straightforward nature of the model—comprising a set of sampled points, called observers—facilitates real-time observability of the current state of the model. This allows for visual inspection of alterations, providing insights into the most relevant parameters and the functionality of SDOstreamclust. For many applications, this capability enables real-time monitoring, which is highly valuable.

Overall, the results affirm that the proposed algorithm is both adaptable and effective in managing concept drift, making it a valuable tool for dynamic clustering scenarios. Its competitive performance across diverse datasets indicates that it is a robust solution for real-time data analysis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Conclusions

5.1 Conclusions

In this work, a data stream clustering algorithm, SDOstreamclust, was developed based on SDOstream [HIZ20], a stream outlier detection algorithm, and SDOclust [IZHZ23], a static clustering method. The design goals for SDOstreamclust include scalability to handle unlimited data streams within resource and time constraints; *adaptability* to effectively adjust to evolving data patterns; *efficiency*, providing a lightweight and performant solution for real-time processing; *interpretability*, offering clear insights into discovered data clusters; an almost *parameter-free* approach that minimizes adjustment needs; and *robustness*, ensuring stability over time with no unexpected performance drifts or high sensitivity to parameter changes. Together, these goals enable SDOstreamclust to effectively address the challenges of dynamic data streams.

The algorithm is implemented in C++ for lightweight and fast performance, and it is wrapped in Python for easy integration and utilization. It is available for use on GitHub [1] or installation via `pip` [2], enhancing usability for practitioners. Its syntax is well-known from scikit-learn, utilizing familiar `fit` and `predict` methods.

The algorithm was evaluated in a diverse test environment, focusing on various datasets, particularly regarding evolving data. It was compared to state-of-the-art alternative algorithms to demonstrate its competitiveness across different scenarios, highlighting the importance of robustness in parameterization. This is crucial, as the algorithm must continue to perform optimally without manual intervention in the face of evolving data. This thorough assessment emphasized the handling of concept drift and evolving data, showcasing the algorithm's adaptability and effectiveness in real-world applications. Furthermore, the analysis of parameterization tested its robustness, confirming that the

[1] `https://github.com/CN-TU/pysdoclust-stream/`
[2] `https://pypi.org/project/pysdoclust-stream/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

algorithm maintains performance under varying conditions with minimal pre-knowledge. Overall, this evaluation aligns with the initially stated research questions, providing valuable insights into the algorithm's capabilities and responsiveness to dynamic data challenges.

The insights gained from this evaluation confirm the algorithm's strengths and provide a clear framework for addressing the research questions that follow.

- **RQ1: Performance**

  How and to what extent can an incremental version of SDOclust improve the performance of consolidated stream clustering in terms of accuracy and execution times?

  SDOstreamclust exhibits competitive performance relative to state-of-the-art alternatives, achieving strong results in accuracy and execution time across a diverse array of synthetic and real-world datasets. This is accomplished with minimal parameter tuning, using an out-of-the-box setup. Detailed evaluation and comparison results can be found in Section 4.1. Furthermore, additional parameter testing in Section 4.2 suggests that even better performance can be attained with little additional effort. While the runtime may not be superior, it remains satisfactory, and the design of SDOstreamclust is highly scalable, making it suitable for speedup through batch processing and buffering if necessary.

- **RQ2: Self-adjustment**

  To what extent can an incremental version of SDOclust effectively self-adjust to overcome the challenges of stream clustering, especially concept drift? Is it superior to state-of-the-art alternatives?

  The incremental SDOstreamclust effectively adapts to various types of evolving data (e.g., concept drift), exhibiting no more difficulty with sudden drift scenarios than with incremental changes. Experiments with specifically designed synthetic data drifts in Section 3.2 demonstrate that SDOstreamclust significantly outperforms most state-of-the-art algorithms. While DBStream remains competitive for many types of concept drift, it falls short in handling sequential sudden drift scenarios.

- **RQ3: Applicability**

  Can interpretability and parameter robustness be maintained in incremental clustering algorithms while ensuring a lightweight profile? To what extent can the models and parameters of SDOstreamclust be considered interpretable and robust?

  Experiments (see Section 4.2) show that a wide range of parameterizations is suitable across all datasets. Three to four key parameters—governing cluster formation, rate of change, and temporal memory—are critical for successful runs and are interpretable regarding their interactions with the data and other parameters. The SDOstreamclust model, composed of sampled points, is inherently simple and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

allows for real-time monitoring. Section 4.3 illustrates this with visualizations further supporting the interpretability of these key parameters and reinforces their explainable nature.

5.2 Future Work

This work has designed and implemented a data stream clustering algorithm tested in a sophisticated environment, successfully meeting its intended characteristics. However, further improvements are necessary and worth pursuing. Possible enhancements related to parameterization are discussed in Section 4.4, while adaptations of parameters during the warm-up phase—when the model is not yet fully grown—are covered in Section 4.3.

We observe that irregular or non-convex cluster shapes require denser representations to accurately capture their complexity compared to simpler, concentric shapes. To enhance representation quality, mechanisms should be considered to prevent sparse cluster representations, particularly for less frequent clusters. In our opinion, preferred sampling alone is insufficient; the method must also ensure that such sampled observers maintain their importance to remain in the model. As representation becomes sparser, a global threshold may be more effective than a local one. Introducing a cluster-global threshold constrained to specific clusters could further enhance representation accuracy.

Moreover, optimizing runtime efficiency is another area worth exploring. For many applications or data streams, the steady recalculation of cutoff thresholds may not be necessary or as critical. Since this operation is rather costly and performed frequently, improving efficiency in this regard could significantly enhance performance. Investigating the trade-offs between this optimization and overall performance will be beneficial.

Furthermore, exploring the trade-offs in parameterization for the related but distinct applications of clustering and outlier detection may yield valuable insights.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Appendix

I Explanation of the Program Code

The core program is written in C++ and builds upon the dSalmon framework, available in the repository[1]. The framework implements the SDOstream algorithm [HIZ20], and we use its M-Tree implementation along with a vector data structure and corresponding metrics. Another third-party library used in our program is the Boost library[2].

The main code files, all contained within the `cpp` folder, are organized as follows:

- `SDOstreamclust.h`: The main header file defining the primary class for the SDOstream clustering implementation. It includes initialization methods, externally callable methods such as fit (and predict) and observer view functions, as well as declarations of all class member functions for core processes. This file serves as the central interface for the SDOstream clustering logic.
- `SDOstreamclust_fitpred.h`: Contains simple internal scripts for managing the fit and/or predict operations. These scripts call member functions for sampling, fitting, graph updating, and predicting, coordinating the overall data processing workflow.
- `SDOstreamclust_sample.h`: Provides functions for sampling or replacing data points from the buffer, essential for maintaining a representative sample set of observers.
- `SDOstreamclust_fit.h`: Contains functions specifically for fitting the model to new incoming data.
- `SDOstreamclust_predict.h`: Implements prediction functions, enabling the assignment of new data points to clusters.
- `SDOstreamclust_observer.h`: Defines a data structure for observers, handling components responsible for monitoring data points and tracking clusters over time.

[1] `https://github.com/CN-TU/dSalmon`

[2] *Boost C++ Libraries*, Version 1.83. Available at: `https://www.boost.org`.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- `SDOstreamclust_cluster.h`: Defines a data structure representing a cluster, detailing the internal characteristics and properties.

- `SDOstreamclust_buffer.h`: Implements a data structure for buffering points temporarily, aiding in efficient data processing.

- `SDOstreamclust_util.h`: Utility functions and helper methods used across different modules, supporting calculations and auxiliary processes.

- `Vector.h` and `util.h`: Provide data structures and utility functions used throughout the clustering framework. These files are sourced from the *dSalmon* repository.

- `MTree.h`, `MTree_impl.h`, and `PlaceholderQueue.h`: Implement the M-Tree data structure, which is used for efficient nearest-neighbor searches in metric spaces. These files are also from the *dSalmon* repository.

- `tpSDOstreamclust` files: These files implement the temporal variation of the SDOstream algorithm, specifically addressing out-of-phase outliers, as described in [HVZ24]. They follow the same modular structure as the standard SDOstreamclust files, with functions tailored to handle time-based adaptations within the clustering process. Although implemented, these files were not analyzed or tested as part of this work.

The `swig` folder contains files for wrapping the C++ implementation with SWIG to make it accessible from Python. The main files include:

- `SDOstreamclust.i`: This is the main SWIG interface file that defines the interface for wrapping the SDOstream clustering implementation with SWIG to a Python module. It includes directives to wrap the relevant C++ headers, including those for methods, array (data structure) types, distance functions, and numpy compatibility.

- `numpy.i`: This file provides the necessary SWIG bindings for numpy, enabling the efficient handling of array types and ensuring smooth integration with numpy arrays in Python.

- `clustering_wrapper.h` and `clustering_wrapper.cxx`: These files define the SWIG wrapper functions for the clustering implementation, enabling the core functionality to be accessed from Python.

- `array_types.h` and `distance_wrappers.h`: These files, sourced from the *dSalmon* repository, provide wrappers for array types and distance calculations, ensuring interoperability between C++ and Python.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The `python` folder contains the Python classes that interface with the SWIG-wrapped C++ implementation. These files import and utilize the SWIG-generated modules and define callable functions such as `fit_predict` and `get_observers`. The two main files in this folder are:

- `clustering.py`: This file defines the `SDOstreamclust` class, which serves as the main Python interface for the SDOstreamclust algorithm. It uses the SWIG-wrapped functions to implement the clustering logic and expose them as Python methods.
- `util.py`: This file handles data preprocessing and cleansing tasks. It provides auxiliary functions that prepare the data for clustering, ensuring that it meets the necessary format and quality for the algorithm.

The `SDOstreamclust` class, implemented in `clustering.py`, serves as an interface to the proposed algorithm.

Initialization: `__init__()`

- **Input:**
  - `k` (*int*, default: 300): Number of observers.
  - `T` (*int*, default: 500): Model's characteristic time.
  - `qv` (*float*, default: 0.3): Ratio of unused observers.
  - `x` (*int*, default: 5): Nearest observers for clustering.
  - `metric` (*str*, default: 'euclidean'): Distance metric.
  - `metric_params` (*dict*): Parameters for the metric.
  - `float_type` (*np.float32/np.float64*, default: `np.float64`): Precision type.
  - `zeta` (*float*, default: 0.6): Ratio for `h`.
  - `chi_min` (*int*, default: 8): Minimum observers for `h`.
  - `chi_prop` (*float*, default: 0.05): Closest observer fraction.
  - `e` (*int*, default: 2): Minimum cluster size.
  - `freq_bins` (*int*, default: 1): Temporal model bins.
  - `max_freq` (*float*, default: 1.0): Temporal model frequency.
  - `outlier_handling` (*bool*, default: False): Outlier handling flag.
  - `rel_outlier_score` (*bool*, default: True): Outlier score type.
  - `outlier_threshold` (*float*, default: 5.0): Outlier threshold.
  - `perturb` (*float*, default: 0.0): Tie-breaking factor.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

  - `random_sampling` (*bool*, default: True): Enables random sampling.
  - `input_buffer` (*int*, default: 0): Batch size for processing.
  - `seed` (*int*, default: 0): Random seed.

- **Functionality:** Validates parameters and initializes the SDOstreamclust algorithm.
- **Output:** Instance of `SDOstreamclust`.

Stream Clustering: `fit_predict()`

- **Input:**
  - `X` (*ndarray*): Input data $(n\_samples, n\_features)$.
  - `times` (*ndarray*, optional): Data timestamps.
- **Functionality:** Processes data, predicts cluster labels, and computes outlier scores.
- **Output:**
  - `labels` (*ndarray*): Cluster labels.
  - `scores` (*ndarray*): Outlier scores.

Observer Retrieval: `get_observers()`

- **Functionality:** Retrieves observer information including positions, labels, and statistics.
- **Output:**
  - `data` (*ndarray*): Observer positions $(n\_observers, n\_features)$.
  - `labels` (*ndarray*): Cluster labels.
  - `observations` (*ndarray*): Moving average of observations.
  - `av_observations` (*ndarray*): Normalized average observations.

Example usage and a demo are available in the repository [3].

[3] https://github.com/CN-TU/pysdoclust-stream/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Overview of Generative AI Tools Used

In this thesis, the generative AI tool ChatGPT (versions GPT-4 and GPT-4o) was utilized to enhance the clarity, coherence, and overall readability of the text. It was employed to refine and reformulate specific sentences, ensuring precise communication while correcting grammar and spelling errors. ChatGPT served as an assistant, aiding in the refinement of the text, but was never used to generate substantial portions of the content.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[AMR$^+$12] Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++ a clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–1, 2012.

[APHW03] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

[Bar02] Daniel Barbará. Requirements for clustering data streams. *ACM sIGKDD Explorations Newsletter*, 3(2):23–27, 2002.

[BFPP06] Christian Böhm, Christos Faloutsos, Jia-Yu Pan, and Claudia Plant. Robust information-theoretic clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 65–75, 2006.

[BH$^+$65] Geoffrey H Ball, David J Hall, et al. *ISODATA, a novel method of data analysis and pattern classification*, volume 699616. Stanford research institute Menlo Park, CA, 1965.

[BKNS00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[CEQZ06] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[CF16] Luis M. Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.

[CPZ+97] Paolo Ciaccia, Marco Patella, Pavel Zezula, et al. M-tree: An efficient access method for similarity search in metric spaces. In *Vldb*, volume 97, pages 426–435. Citeseer, 1997.

[CT07] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2007.

[CT18] Matthias Carnein and Heike Trautmann. evostream–evolutionary stream clustering utilizing idle times. *Big data research*, 14:101–111, 2018.

[CT19] Matthias Carnein and Heike Trautmann. Optimizing data stream representation: An extensive survey on stream clustering algorithms. *Business & Information Systems Engineering*, 61:277–297, 2019.

[DSMW20] Leonardo Enzo Brito Da Silva, Niklas Max Melton, and Donald C Wunsch. Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study. *IEEE Access*, 8:22025–22047, 2020.

[Edg87] Francis Ysidro Edgeworth. Xli. on discordant observations. *The london, edinburgh, and dublin philosophical magazine and journal of science*, 23(143):364–375, 1887.

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[FGG+18] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*, volume 10. Springer, 2018.

[GMMO00] S Guha, N Mishra, R Motwani, and L O'Callaghan. Clustering data streams. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE, 2000.

[GMRS16] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721. PMLR, 2016.

[GŽB+14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[Har23] Alexander Hartl. *Anomaly Detection for Network Security based on Streaming Data*. PhD thesis, Technische Universität Wien, 2023.

[Haw80] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[HB16] Michael Hahsler and Matthew Bolaños. Clustering data streams based on shared density between micro-clusters. *IEEE transactions on knowledge and data engineering*, 28(6):1449–1461, 2016.

[HHH$^+$22] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35:32142–32159, 2022.

[HIZ20] Alexander Hartl, Félix Iglesias, and Tanja Zseby. Sdostream: Low-density models for streaming outlier detection. In *ESANN*, pages 661–666, 2020.

[HLA23] Dania Herzalla, Willian T Lunardi, and Martin Andreoni. Tii-ssrc-23 dataset: Typological exploration of diverse traffic patterns for intrusion detection. *IEEE Access*, 2023.

[HS00] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information processing letters*, 76(4-6):175–181, 2000.

[HVZ24] Alexander Hartl, Félix Iglesias Vázquez, and Tanja Zseby. Sdooop: Capturing periodical patterns and out-of-phase anomalies in streaming data analysis. *arXiv preprint arXiv:2409.02973*, 2024.

[HYS$^+$17] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.

[Igl21] F Iglesias. Data for evaluation of stream data analysis algorithms. *Mendeley Data*, 10:c43kr4t7h8, 2021.

[IMZ24] Félix Iglesias, Conrado Martínez, and Tanja Zseby. Impact of the neighborhood parameter on outlier detection algorithms. In *International Conference on Similarity Search and Applications*, pages 88–96. Springer, 2024.

[IOHZ20] Félix Iglesias, Denis Ojdanic, Alexander Hartl, and Tanja Zseby. Mdcstream: Stream data generator for testing analysis algorithms. In *Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 56–63, 2020.

[IV24] F. Iglesias Vazquez. Sdostreamclust: Stream clustering robust to concept drift - evaluation tests (1.0.0). `https://doi.org/10.48436/xh0w2-q5x18`, 2024.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[IVZ23] Félix Iglesias Vázquez and Tanja Zseby. Temporal silhouette: validation of stream clustering robust to concept drift. *Mach. Learn.*, 113(4):2067–2091, nov 2023.

[IZHZ23] Félix Iglesias, Tanja Zseby, Alexander Hartl, and Arthur Zimek. Sdoclust: Clustering with sparse data observers. In *International Conference on Similarity Search and Applications*, pages 185–199. Springer, 2023.

[JM06] Han Jiawei and Kamber Micheline. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[Lug08] Edwin Lughofer. Extensions of vector quantization for incremental clustering. *Pattern recognition*, 41(3):995–1011, 2008.

[M$^+$67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[MHM$^+$21] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110):1–8, 2021.

[MLA18] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1963–1972, 2018.

[MTRAR$^+$12] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[NN19] Olfa Nasraoui and C-E Ben N'Cir. Clustering methods for big data analytics. *Techniques, Toolboxes and Applications*, 1:91–113, 2019.

[NWN15] Hai-Long Nguyen, Yew-Kwong Woon, and Wee-Keong Ng. A survey on data stream clustering and classification. *Knowledge and information systems*, 45:535–569, 2015.

[OMM$^+$02] Liadan O'callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*, pages 685–694. IEEE, 2002.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[Pev16] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2016.

[PGR18] Divya Pandove, Shivan Goel, and Rinkl Rani. Systematic review of clustering high-dimensional and large datasets. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):1–68, 2018.

[Ran71] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[RKV$^+$21] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[RM09] Jiadong Ren and Ruiqing Ma. Density-based data streams clustering over sliding windows. In *2009 Sixth international conference on fuzzy systems and knowledge discovery*, volume 5, pages 248–252. IEEE, 2009.

[Rou87] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[SA16] Saket Sathe and Charu C Aggarwal. Subspace outlier detection in linear time with randomized hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 459–468. IEEE, 2016.

[SFB$^+$13] Jonathan A Silva, Elaine R Faria, Rodrigo C Barros, Eduardo R Hruschka, André CPLF de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31, 2013.

[Ste56] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bulletin de l'Académie Polonaise des Sciences*, 1:801–804, 1956.

[VHZZ23] Félix Iglesias Vázquez, Alexander Hartl, Tanja Zseby, and Arthur Zimek. Anomaly detection in streaming data: A comparison and evaluation study. *Expert Systems with Applications*, 233:120994, 2023.

[VZZ18] Félix Iglesias Vázquez, Tanja Zseby, and Arthur Zimek. Outlier detection based on low density models. In *2018 IEEE international conference on data mining workshops (ICDMW)*, pages 970–979. IEEE, 2018.

[Wat23] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[WY05] Kuo-Lung Wu and Miin-Shen Yang. A cluster validity index for fuzzy clustering. *pattern recognition letters*, 26(9):1275–1291, 2005.

[ZCQJ08] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15:181–214, 2008.

[ZCY$^+$06] Aoying Zhou, Feng Cao, Ying Yan, Chaofeng Sha, and Xiaofeng He. Distributed data stream clustering: A fast em-based approach. In *2007 IEEE 23rd international conference on data engineering*, pages 736–745. IEEE, 2006.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.