International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
Article Received: *25 September 2023* ***Revised:*** *12 November 2023* ***Accepted:*** *30 November 2023*

ClaraStream: A Novel Algorithm for Real-Time Data Stream Clustering

[1]Ms. Neha Sharma , [2]Dr. Shradhdha Masih
[1]Assistant Professor , Sage university, Indore
[2]Associate Professor , SCSIT, DAVV Indore

Abstract**:** One crucial step in extracting knowledge from datasets is to cluster or split the records of the data set into groups of related records. The detection of clusters in very large, multi-dimensional, static datasets has been the subject of extensive investigation. Unfortunately, this study has led to the development of classical clustering that is ineffective for clustering data streams. A data stream is a dynamic data set that is defined as an infinite sequence of data records that changes over time and arrives at very fast rates. There are many processes in the world today that produce rapidly changing data streams at high speeds. Credit card transactions, click streams, and sensor networks are a few examples. The rapid proliferation of data in various fields necessitates the development of algorithms capable of processing and analyzing data streams in real-time. ClaraStream is a novel clustering algorithm designed to efficiently handle the unique challenges posed by data streams, including their high volume, velocity, and potentially boundless nature. Unlike traditional clustering methods that are suitable for static datasets, ClaraStream offers a two-phase approach—online micro-clustering and offline macro-clustering—that enables real-time processing and trend analysis. This paper provides a comprehensive overview of the ClaraStream algorithm, its architecture, and its application to air quality data streams.

Keywords : ClaraStream, Stream Clustering

1. Introduction

The digital landscape has experienced exponential growth in recent years. According to projections by the International Data Corporation (IDC), the digital universe was expected to expand from 0.18 zettabytes in 2006 to a staggering 1.8 zettabytes by 2011 - a tenfold increase. A significant contributor to this data explosion is the proliferation of high-speed, ever-evolving data streams. In response to this data deluge, clustering analysis - a data mining technique has been increasingly applied to streaming data. However, the unique characteristics of data streams, including their unbounded nature, constant evolution, and variable high-speed influx, pose significant challenges to traditional clustering methods.

Clustering is a fundamental task in data analysis, involving the partitioning of data into distinct groups based on similarity. However, when dealing with data streams, traditional clustering methods fall short due to the continuous and rapid influx of data. Data streams are characterized by their high volume, velocity, and unbounded nature, which introduces the challenge of real-time processing and analysis. In this context, ClaraStream emerges as a robust solution, designed specifically to address these challenges by employing a distributed and scalable architecture that facilitates efficient data stream clustering.

2. Background

2.1. Data Stream Clustering

Data stream clustering involves the real-time partitioning of data that arrives continuously over time. The key challenges in this domain include managing the high arrival rate of data, handling concept drift (changes in the data distribution over time), and ensuring scalability and efficiency. Traditional clustering algorithms, such as k-means and hierarchical clustering, are not suitable for data streams due to their assumption of static datasets and multiple-pass processing.

Data streams represent a continuous flow of structured information, characterized by their rapid influx, chronological arrangement, dynamic nature, and limitless scope. These streams typically generate data at an impressive rate, often surpassing thousands of entries per second. As time progresses, the underlying patterns and insights derived from these streams undergo shifts, a phenomenon known as concept evolution. Given their

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 September 2023 **Revised:** 12 November 2023 **Accepted:** 30 November 2023*

unending nature, traditional storage methods like database management systems prove inadequate for retaining the entirety of a data stream's content.

The fluidity of data streams manifests in various ways, including fluctuating statistical distributions, shifting data groupings, and alterations in categorical or feature-based classifications. Real-world examples of such streams are abundant, encompassing internet traffic flows, data from sensor arrays, wireless communication networks, RFID systems, user interaction logs on websites, and telecommunications data. For the purposes of this research, we define a data stream S as an infinite series of dual elements (s, t). In this representation, 's' denotes a structured data entry containing multiple attributes, while 't' represents a timestamp automatically assigned by the system at the moment of the data entry's creation. This timestamp 't' serves as the stream's unique identifier, with values that consistently increase over time. It's important to note that the timestamp values in one data stream operate independently from those in other streams processed within the ClaraStream Algorithm. Given that our focus is on structured data records, it's crucial to clarify that this definition excludes streams of unstructured information, such as continuous audio or video feeds. These types of streams, while valuable in other contexts, fall outside the scope of our current discussion on data streams. The structured nature of these data streams allows for systematic analysis and processing, which is central to the approach described in this paper. Each data record within the stream contains predefined attributes, enabling efficient handling and interpretation of the incoming information. With the exponential growth of data generated from various sources like social media, IoT devices, and financial transactions, there is a pressing need for algorithms that can efficiently process and cluster data streams. ClaraStream aims to fulfill this need by providing a scalable and flexible clustering solution that operates in real-time, handling high-dimensional data with ease.

3. Related Work

The field of clustering large-scale, multi-dimensional datasets has attracted significant scholarly attention. A primary focus of this research has been the development of clustering algorithms that can operate efficiently within the constraints imposed by massive datasets, particularly in terms of memory usage and processing time relative to data volume.These challenges are significantly intensified in the context of data streams, which are characterized by their rapid inflow and infinite nature. The continuous, high-speed arrival of data points in streams pushes the boundaries of traditional clustering methods, necessitating novel approaches.Several groundbreaking studies have made substantial contributions to addressing these challenges. Notably, the research presented in three key papers introduced innovative concepts, data structures, and algorithmic approaches that have significantly advanced our ability to cluster streaming data efficiently. Building upon this foundation, the ClaraStream algorithm represents a further evolution in stream clustering techniques. It incorporates and extends the principles established in these seminal works, introducing new methodologies to enhance performance and adaptability in stream processing scenarios. The ClaraStream approach aims to strike a balance between computational efficiency and clustering accuracy, taking into account the unique characteristics of data streams. By leveraging and expanding on previous research, it seeks to provide a robust solution for real-time clustering of continuous, high-volume data streams.

BIRCH

The BIRCH algorithm is a hierarchical clustering method designed for large datasets. It operates by constructing a tree-like data structure called the Clustering Feature Tree (CFT). This approach allows for efficient memory usage and speedy processing of vast amounts of data.

Clustering Feature (CF): A CF is a compact representation of a cluster, defined by a triple:

$$CF = (N, LS, SS)$$

Where:

- N: Number of data points in the cluster
- LS: Linear sum of the N data points $LS = \sum_{\{i=1\}}^{\{N\}} x\_i$
- SS: Square sum of the N data points $SS = \sum_{\{i=1\}}^{\{N\}} |x_i|^2$

Clustering Feature Tree (CFT): A height-balanced tree with two parameters:

- B: Maximum number of entries in each non-leaf node
- L: Maximum number of entries in each leaf node

Mathematical Foundations:

1. Centroid calculation: Centroid $= \frac{LS}{N}$
2. Radius (R) of a cluster: $R = \sqrt{\frac{SS}{N} - \left\{\frac{LS}{N}\right\}^2}$

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 September 2023 **Revised:** 12 November 2023 **Accepted:** 30 November 2023*

3. Distance between clusters (D): For two CFs, CF1 and CF2: D = $sqrt[\frac{LS1}{N1} - \left\{\frac{LS2}{N2}\right\}^2]$

The BIRCH algorithm's ability to handle large datasets efficiently makes it particularly useful in scenarios where data volume is a significant concern. Its hierarchical approach allows for multi-level analysis of the clustering structure.

CluStream

CluStream is a popular algorithm for clustering evolving data streams. It effectively handles large and continuously arriving data by maintaining a summary of the data using micro-clusters and periodically applying macro-clustering techniques to obtain the final clusters.

Micro-clusters: These are compact data summaries that represent dense regions of the data stream over time. Each micro-cluster stores information about a small group of data points. As data points arrive in the stream, they are absorbed into existing micro-clusters or used to create new ones. Each micro-cluster is defined by a set of statistics that summarize the data points within it.

Each micro-cluster is represented by a tuple:

$$MC = (N, LS, SS, TLS, TSS)$$

Where:

- N: The number of data points in the micro-cluster.
- LS: The linear sum of the data points in the micro-cluster.
- SS: The squared sum of the data points in the micro-cluster.
- TLS: The linear sum of the timestamps associated with the data points.
- TSS: The squared sum of the timestamps.

From this, we can compute:

- **Centroid** C $= \frac{LS}{N}$
- **Radius** of the micro-cluster: $R = \sqrt{\frac{SS}{N} - \left\{\frac{LS}{N}\right\}^2}$
- **Time Stamp** of the micro-cluster: Tc = $\frac{TLS}{N}$
- **Time Deviation**: σT = $\sqrt{\frac{TSS}{N} - \left\{\frac{TLS}{N}\right\}^2}$

Decay Function

CluStream employs a decay function to give more importance to recent data points. The decay factor is often modeled as an exponential decay:

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

Where:

- tcurrent is the current time.
- t is the time of a given data point.
- λ is the decay rate.

Online Micro-Cluster Maintenance

As new data points arrive:

1. Compute the distance of the new point to all existing micro-clusters.
2. If the point is within the radius of a micro-cluster, update that micro-cluster:

N=N + 1

LS=LS + x

SS=SS + x^2

TLS=TLS + t

TSS=TSS + t^2

3. If the point is not close enough to any micro-cluster, a new micro-cluster is created.
4. Periodically, outdated micro-clusters are discarded based on the decay function.

Pyramidal time frame is a concept used to manage and store the history of micro-clusters at different time resolutions, enabling efficient retrieval and analysis of clustering information over varying time horizons. CluStream considers the entire history of the data stream by using a time horizon, which can be defined as the period over which the clustering is performed.

Users can perform temporal queries on the data stream using the pyramidal time frame. For example, they might query the state of clusters over the last hour with a minute-by-minute resolution or over the last day with an hourly resolution.

Snapshots: At regular intervals, CluStream stores snapshots of the micro-clusters. These snapshots help in tracking the evolution of clusters over time.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
Article Received: *25 September 2023* ***Revised:*** *12 November 2023* ***Accepted:*** *30 November 2023*

Offline Macro-Clustering

At specified intervals, the micro-clusters are subjected to traditional clustering methods like K-means to form macro-clusters, which are the final output clusters.The offline phase aims to generate a higher-quality clustering of the data stream using the macroclusters as pseudo-points. This phase is typically executed on-demand when a user requests clustering results.The input consists of q macroclusters, each representing a summary of data points from different time horizons.After obtaining the final clusters, various metrics can be used to evaluate cluster quality.

4. ClaraStream Algorithm

Overview

ClaraStream is designed to perform clustering on data streams in a single pass, making it suitable for real-time applications. The CluSandra algorithm presents an innovative approach to data stream clustering, integrating temporal and spatial aspects. The algorithm is divided into two main phases: online micro clustering and offline macro clustering.

Parallel Stream Data collection

Web scraping is a powerful technique for extracting information from websites using automated tools or scripts. This method is widely employed in data extraction tasks, where information found on the web needs to be collected and analyzed. The process typically begins with unstructured data, often in HTML format, which is then transformed into a structured format like a spreadsheet or database for easier manipulation and analysis. Web scraping can be accomplished through various means, including online services, specific APIs, or custom-coded solutions tailored to the task at hand. At the heart of web scraping are two key components: the crawler and the scraper.

The crawler, an AI-driven algorithm, acts as the navigator, systematically browsing the web to locate the required data. It traverses through interconnected links across the internet, identifying relevant pages for data extraction. The scraper, on the other hand, is a specialized tool designed to extract the desired information from these identified web pages. It employs various techniques to efficiently pull out specific data points from the HTML structure of the page. For our work we are using Air Quality Data set from thingspeak.com, it provide live data of air quality of different zone. Web scraper is used to access parallel data of different zone. The dataset shown below:

	created_at	entry_id	Ozon ppb	CO ppm	Oxygen ppm	Sound Level DB	Dust Concentration	Tempera ture	% Humidity
2	2024-01-15 16:05:48 UTC	2289071	50.1	125	0.8	11.8	984.7	20.5	50.6
3	2024-01-15 16:07:52 UTC	2289072	50.7	125.6	0.8	21.3	656.3	20.1	50.7
4	2024-01-15 16:09:53 UTC	2289073	51	125.1	0.8	16.5	1037.1	20.5	51
5	2024-01-15 16:11:54 UTC	2289074	51.3	124.8	0.8	26.8	849.9	20.7	51
6	2024-01-15 16:13:56 UTC	2289075	50.4	124.1	0.8	15.8	920.5	20.5	50.8
7	2024-01-15 16:15:57 UTC	2289076	49.9	123.6	0.8	26.8	686.6	20.5	50.8
8	2024-01-15 16:17:59 UTC	2289077	49.9	123.3	0.8	24.2	704.8	20.3	50.4
9	2024-01-15 16:20:02 UTC	2289078	49.7	123.7	0.8	23.4	645.2	19.9	50.4
10	2024-01-15 16:22:06 UTC	2289079	49.8	122.8	0.8	16.8	1154.7	20.3	51.3
11	2024-01-15 16:24:11 UTC	2289080	50.2	122.7	0.8	43.2	809.3	20.3	51.1
12	2024-01-15 16:26:11 UTC	2289081	49.7	122.5	0.8	14.8	1187.4	20.3	51
13	2024-01-15 16:28:13 UTC	2289082	49.9	123	0.8	20.1	556.6	19.9	50.8
14	2024-01-15 16:30:18 UTC	2289083	49.9	123.3	0.8	19.5	916.5	19.9	51.2
15	2024-01-15 16:32:21 UTC	2289084	50.7	123.8	0.8	31	332.3	19.9	51.5
16	2024-01-15 16:34:23 UTC	2289085	50.7	124	0.8	23.7	925.9	19.9	51.8
17	2024-01-15 16:36:24 UTC	2289086	51.1	123.8	0.8	19.4	844.7	19.9	51.7
18	2024-01-15 16:38:27 UTC	2289087	49.4	123.5	0.8	12.4	722.7	19.8	51.4
19	2024-01-15 16:40:28 UTC	2289088	49.7	122.5	0.8	21.8	1069.1	19.8	51.1
20	2024-01-15 16:42:30 UTC	2289089	50.5	123.1	0.8	21.3	940.5	19.6	51
21	2024-01-15 16:44:32 UTC	2289090	49.8	123	0.8	27	897.2	19.4	51
22	2024-01-15 16:46:36 UTC	2289091	49.8	123.1	0.8	19.9	757.7	19.2	50.6

Fig1:Air Quality Data set

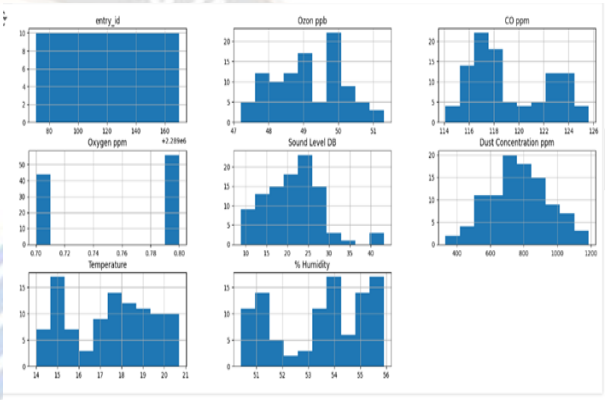

Fig.2: Histogram view of Dataset

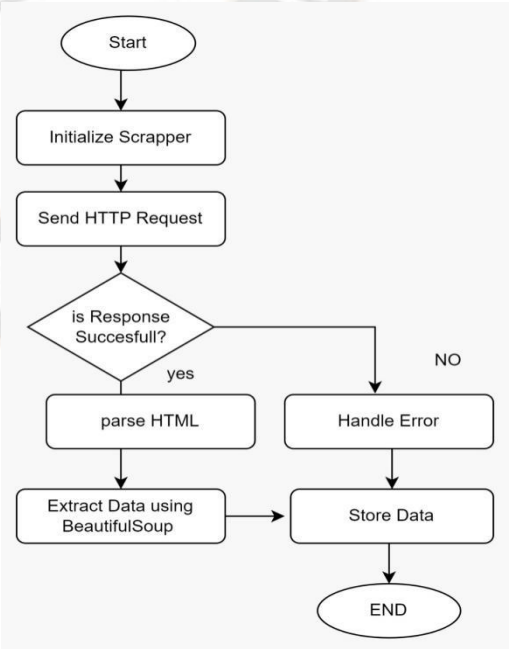

Fig. 3: Web scraper data collection

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 September 2023 **Revised:** 12 November 2023 **Accepted:** 30 November 2023*

Online Micro-Clustering

The microcluster formation in the online phase of the ClaraStream algorithm involves the real-time processing of incoming data points to create and update microclusters. These microclusters are compact summaries of the data and are continuously adjusted as new data arrives. Each data point is evaluated based on its Euclidean distance to the centroids of existing micro-clusters. If the data point is close to an existing micro-cluster, it is absorbed into that cluster, and the cluster's summary statistics are updated. If no suitable micro-cluster exists, a new one is created. This approach ensures that the algorithm adapts to changes in the data stream over time.

The CluStream and ClaraStream algorithms differ significantly in their approach to managing microclusters. CluStream employs a fixed set of microclusters, denoted as M = {M1, M2, …, Mq}, which it creates and maintains in memory. This set has a maximum size q, with each microcluster assigned a unique identifier. However, CluStream's performance is notably sensitive to any additions beyond this fixed number, as exceeding q negatively impacts the algorithm's efficiency. This limitation creates a significant space constraint for CluStream. In contrast, ClaraStream takes a more flexible approach. While it also maintains a working set of microclusters in memory, the size of this set is not rigidly fixed. Instead, ClaraStream determines the number of microclusters based on two dynamic factors: a sliding time window and a maximum boundary threshold (MBT).

Cluster Feature (CF) Vector: Each microcluster in ClaraStream is represented by a CF vector, which summarizes the data points within the cluster. The CF vector consists of three components. Mathematically, for a microcluster C the CF vector is defined as:

CF = (N, LS, SS, TLS, TSS, CT, LAT)

- **N**: Number of data points in the microcluster.
- **LS**: Linear sum of the data points (i.e., the sum of all the data points in the microcluster).
- **SS**: Squared sum of the data points (i.e., the sum of the squares of all the data points in the microcluster).
- **TLS**: The linear sum of the timestamps associated with the data points.
- **TSS**: The squared sum of the timestamps
- **CT:** Creation Time of Microlcuster
- **LAT:** Last data point absorption time of a Microcluster

Centroid of the Microcluster: The centroid μi of the microcluster Ci can be calculated using the linear sum LSi and the number of points Ni

$$\mu i = \frac{\text{LSi}}{Ni}$$

Radius of the Microcluster: The radius Ri of a microcluster, which measures the spread of the data points within the cluster, is computed as:

$$R = \sqrt{\frac{SSi}{N\,i} - \left\{\frac{LSi}{Ni}\right\}^2}$$

Distance Calculation: When a new data point xt arrives, the algorithm calculates its distance to the centroid of each existing microcluster Ci. The distance d(xt,μi) is typically calculated using the Euclidean distance formula:

$$\boldsymbol{d(x_t, \mu_i) = \sqrt{\{(x_t - \mu_i) \cdot (x_t - \mu_i)\}}}$$

The microcluster Mi with the smallest distance d(xt,μi) is selected as the closest cluster.

Assignment to Microcluster: The data point xt is assigned to the nearest microcluster Ci if the distance d(xt,μi) is less than a predefined threshold ϵ. This ensures that the data point is similar enough to be part of the existing cluster.

Updating Microcluster: If xt is assigned to microcluster Ci, the CF vector of Ci is updated as follows:

Ni←Ni+1

LSi←LSi+xt

SSi←SSi+xt^2

The centroid μi and radius Ri of the microcluster are recalculated using the updated CF vector.

Instead of calculating the maximum boundary threshold (MBT) based on the RMSD or nearest neighbor distance for microclusters with only one data point, ClaraStream uses a configurable, fixed maximum radius to determine whether a microcluster can absorb a new data record. This simplifies the decision process and provides more consistent behavior. Unlike CluStream, which may merge or delete microclusters to conserve memory, ClaraStream simply adds a new microcluster when an existing one cannot absorb a new data record. However, ClaraStream in-memory working set is managed using a temporal sliding window, where microclusters that are no longer active within the window are removed but persisted to a database. ClaraStream uses a

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 September 2023 **Revised:** 12 November 2023 **Accepted:** 30 November 2023*

sliding window to manage the size of its in-memory microclusters. As time progresses, older microclusters that fall outside the window are removed from memory, but they are not lost, instead, they are saved in a persistent storage system. Suppose a new data point xt = [3,4] arrives, and the closest microcluster Mi has a centroid μi=[2,3] with an RMSD of 1. If the factor t=2, the MBT for Mi would be:

$$MBT=2\times1=2$$

If the distance from xt to μi is less than 2, Mi absorbs xt. If not, a new microcluster is created for xt. In ClaraStream, if the microcluster's radius is set to a fixed value (say 2.5), the algorithm checks if the distance is less than 2.5 before absorption. If it's larger, a new microcluster is added, with no merging or deletion until the temporal window dictates it.

These mechanisms ensure that both algorithms manage the dynamic nature of data streams effectively, with ClaraStream offering more straightforward and scalable management, especially in distributed environments. In data stream analysis, patterns may emerge, vanish, and then resurface over time.

This phenomenon is captured by the creation of microclusters with similar spatial characteristics but distinct temporal attributes. The lifespan of a cluster can be determined using Creation Time (CT) and Last Active Time (LAT) parameters, allowing for detailed statistical analysis of both temporal and spatial aspects. This analysis can include examining the temporal density of data records and their spatial distribution relative to each other or their centroid.

When a microcluster becomes inactive, it can no longer absorb new data records. However, it remains valuable during the macroclustering phase, where it can be combined with other inactive or active microclusters to form larger superclusters. To manage the continuous influx of data in an unbounded stream, the algorithm's microclustering phase operates within a defined temporal sliding window.

This window approach ensures that only microclusters within the current time frame are updated, allowing the system to adapt to evolving patterns while maintaining computational efficiency. This method effectively balances the need to capture historical patterns with the requirement to process new, incoming data in real-time.

Offline Macro-Clustering

The offline phase involves aggregating micro-clusters into macro-clusters based on temporal and spatial criteria. This phase allows for the identification of long-term trends and patterns in the data stream. The CF vectors of the micro-clusters are used to efficiently perform this aggregation, enabling the detection of both spatial and temporal patterns in the data. The Macrocluster are evaluated using CLARASTREAM on microclsuters.

5. Empirical Result

The evaluation process for the ClaraStream algorithm involved a series of experiments designed to assess its accuracy. These tests were conducted on a relatively modest scale, utilizing a algorithm that consisted of a small cluster of two to three compute nodes. This setup, while sufficient for initial validation, represents a controlled environment that allows for focused analysis of the algorithm's performance.

The choice of a small-scale testing environment likely allowed researchers to closely monitor the algorithm's behavior and make precise observations about its accuracy under controlled conditions. Such an approach is common in the early stages of algorithm development and validation, as it allows for easier troubleshooting and fine-tuning. Performances of clusters are measured by similarity between data points of a single clusters and dissimilarity between data points of different clusters. Silhouette Score and Adjusted Rand Index (ARI) are two most popular for measuring clusters performance.

Silhouette Score

Silhouette coefficient or score(s) is evaluation matrices, bounded between -1 to +1. Here -1 indicates incorrect clustering and +1 indicates highly dense clustering. Score around zero indicate overlapping in clusters.

Silhouette score measures closeness of data points within cluster. Silhouette score measures closeness of data points within cluster.

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

Where

a = The mean/average distance between a data point and all other points in the same cluster.This measures how closely related the data point is to other points in the same cluster. A smaller value of a indicates that the point is well-matched with its cluster.

b = The mean/average distance between a data point and all other points in the next nearest cluster.This measures how far the data point is from the points in the nearest cluster that

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11
Article Received: *25 September 2023* ***Revised:*** *12 November 2023* ***Accepted:*** *30 November 2023*

it is not a part of. Larger value of b indicates that the point is well-separated from the other clusters.

The Silhouette Score is a powerful tool for evaluating clustering results, helping to ensure that clusters are well-formed and distinct from one another. It's particularly useful in choosing the right number of clusters and comparing different clustering methods.

Adjusted Rand Index

The Adjusted Rand Index (ARI) is a metric used to evaluate the similarity between two data clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings.

The ARI is a corrected-for-chance version of the Rand Index, a fundamental measure of cluster similarity. The ARI adjusts the Rand Index (RI) to account for the chance grouping of elements, providing a more accurate measure of the agreement between two clusterings.

The Rand Index measures the percentage of decisions that are correct (i.e., pairs of elements that are either clustered together or separately in both the true labels and the clustering algorithm's output).

The ARI improves upon this by adjusting the score based on the expected similarity of all random clusterings, meaning it compensates for the fact that random clusterings could appear to be similar just by chance.

Adjusted Rand Index (ARI) is used to measure similarity between clusters. ARI value 1 indicates identical cluster, -0.5 indicates discordant cluster. ARI is similar to Random index (RI), instead of RI the result of ARI lies between -1 to +1. ARI of cluster are measure by formula

$$ARI = (RI - expected) / (RImaxRI - expectedRI)$$

Where RI = stand for rand index, calculates measure between each two data point pairs with in cluster.

S. No.	Clustering Methods	Silhouette Score	Adjusted Rand Index (ARI)
1	BIRCH	0.457	0.501
2	CLUSTREAM	0.487	0.459
3	CLARASTREAM	0.493	0.606

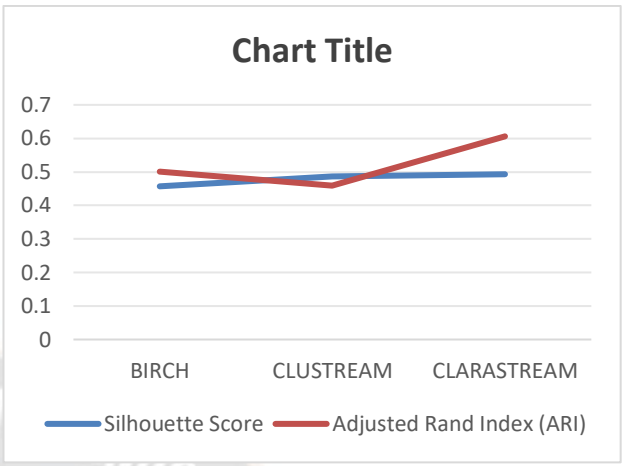

Fig.4: Comparison of Algorithms

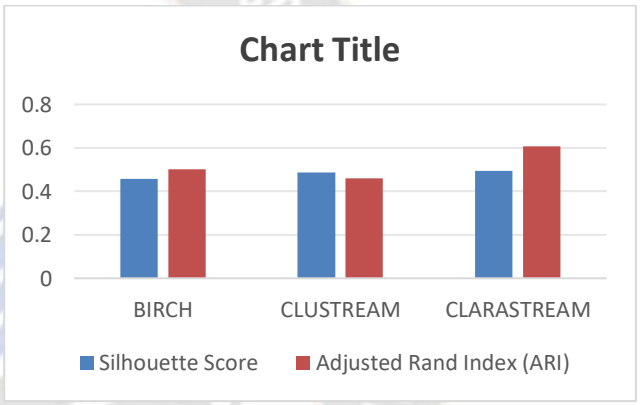

Fig.5: Comparison of Algorithms

6. Conclusion

The ClaraStream algorithm presents an innovative approach to clustering evolving data streams, offering significant improvements over existing methods that attempt to cluster entire streams simultaneously. By treating the stream as a dynamic, ever-changing entity, ClaraStream provides a more nuanced and adaptable clustering solution. At its core, ClaraStream employs a dual-component strategy: a real-time statistical data gatherer and a separate analytical processor. This design enables the algorithm to characterize stream clusters across various temporal and spatial scales, granting analysts unprecedented flexibility in examining data patterns as they emerge and evolve.

Moreover, ClaraStream introduces the concept of microclustering, which allows for more detailed data representation compared to traditional methods like STREAM. By maintaining a richer set of statistics rather than just k points used in k-means approaches, ClaraStream

achieves superior clustering accuracy while maintaining scalability.

The microclustering technique strikes a balance between comprehensive data collection and efficient processing, ensuring that sufficient information is preserved for effective cluster analysis without overwhelming system resources. This makes ClaraStream particularly well-suited for handling large-scale, rapidly changing data streams in real-world applications.

ClaraStream represents a significant advancement in the field of data stream clustering, offering a scalable and efficient solution for real-time data analysis. Its two-phase approach, combined with a distributed architecture, makes it well-suited for modern applications where data streams are continuous and voluminous. Future work will explore further optimizations and potential applications in other domains.

References

[1] Aggarwal, Charu. (2004). A Framework for Diagnosing Changes in Evolving Data Streams. Proceedings of the ACM SIGMOD International Conference on Management of Data. 575-586. 10.1145/872757.872826. .

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. SIGMOD Rec. 28, 2 (June 1999), 49–60. https://doi.org/10.1145/304181.304187.

[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '02). Association for Computing Machinery, New York, NY, USA, 1–16.

[4] P. S. Bradley, Usama Fayyad, and Cory Reina. 1998. Scaling clustering algorithms to large databases. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98). AAAI Press, 9–15.

[5] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. 2002. Mining data streams under block evolution. SIGKDD Explor. Newsl. 3, 2 (January 2002), 1–10. https://doi.org/10.1145/507515.507517

[6] Ji Zhang, Qigang Gao, Hai Wang, Qing Liu, and Kai Xu. 2009. Detecting Projected Outliers in High-Dimensional Data Streams. In Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA '09). Springer-Verlag, Berlin, Heidelberg, 629–644. https://doi.org/10.1007/978-3-642-03573-9_53

[7] C. Plant, C. Bohm, Novel Trends In Clustering, Technische Universitat, Munchen Germany, Ludwig Maximialns Universitat Munchen, Germany, 2008.

[8] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. SIGMOD Rec. 25, 2 (June 1996), 103–114. https://doi.org/10.1145/235968.233324

[9] Kuncheva, Ludmila. (2008). Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. Proc. Eur. Conf. Artif. Intell..

[10] Ji Zhang, Qigang Gao, Hai Wang, Qing Liu, and Kai Xu. 2009. Detecting Projected Outliers in High-Dimensional Data Streams. In Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA '09). Springer-Verlag, Berlin, Heidelberg, 629–644. https://doi.org/10.1007/978-3-642-03573-9_53