An Incremental Clustering Algorithm for Arbitrary Shaped in IoT Data

Tianzhen Chen[1,2,*]

[1]EIT Data Science and Communication College, Zhejiang Yuexiu University of Foreign Languages, Shaoxing, China
[2]School of Information Industry, Meizhouwan Vocational Technical College, Putian, China
**Corresponding author: terrychen17@outlook.com*

Abstract: *With the rapid development of Internet of Things (IoT) technology, a vast number of IoT devices continuously generate large and ever-growing datasets, which typically exhibit high-dimensional and dynamically changing characteristics. To effectively manage these data, this study introduces a novel incremental clustering algorithm for arbitrary shaped in IoT data named Arbitrary Shaped Incremental Clustering (ASIC). The ASIC algorithm adapts to ongoing changes in data distributions by updating clustering results in real-time, thus effectively managing dynamic data without the need to reprocess the entire dataset. By comparing with several existing clustering algorithms, this study demonstrates the advantages of ASIC in handling large-scale and dynamic datasets. Experiments conducted on multiple datasets, including AWS IoT, show that the ASIC algorithm excels in clustering quality and operational efficiency. This algorithm is particularly suitable for applications requiring rapid response to environmental changes and real-time updates of data processing results.*

Keywords: *Data Mining, IoT Data, Incremental Learning, Clustering Analysis, Data Analysis*

1. Introduction

In the modern era, characterized by rapid technological advancements, the proliferation of Internet of Things (IoT) devices has catalyzed an unprecedented generation of voluminous and complex datasets [1]. These datasets are inherently dynamic, perpetually expanding, and feature volatile characteristics that challenge traditional data management strategies. Conventional batch processing methods, which necessitate the reprocessing of entire datasets following any update, prove inadequate in this context due to their inability to facilitate real-time processing and to optimize computational efficiency [2]. Therefore, the development of incremental clustering methods, which can continuously update clustering outcomes as new data flows in, becomes indispensable. Such methods not only accommodate the continuous data streams but also maintain the agility necessary for real-time processing and adapt dynamically to changes in data characteristics, thus significantly mitigating computational resource consumption [3].

This paper is propelled by the necessity to refine IoT data handling through a novel incremental clustering algorithm for arbitrary shaped in IoT data known as Arbitrary Shaped Incremental Clustering (ASIC). ASIC is a significant enhancement over the existing online arbitrary shaped clustering (OASC) methodology, designed to more adeptly process data distributions of arbitrary shapes. By integrating a batch processing mechanism, ASIC not only accelerates the speed of data processing but also improves the adaptability and precision of clustering across diverse datasets. This research substantiates the efficacy of ASIC through rigorous empirical evaluations across multiple datasets, which illuminate its superior performance in terms of execution speed, clustering accuracy, and adaptability to incremental clustering scenarios. The primary contribution of this research lies in its potential to furnish an efficient and robust framework for real-time, large-scale data analysis within the IoT landscape, thereby augmenting the decision-support capabilities and operational efficiency of IoT systems.

1.1 Research significance

The significance of this research is manifold, extending beyond the technical enhancements it introduces. First, it addresses a critical gap in the literature concerning efficient data processing methods suited for the high-volume, dynamic nature of IoT-generated data. Second, it offers a scalable solution that could be pivotal for real-time applications across various sectors including healthcare, urban intelligence, and industrial automation. These applications often require immediate data processing and responsiveness to environmental changes, which are capabilities ASIC is specifically designed to deliver.

1.2 Objectives

The objectives of this research are to:

Demonstrate the superiority of the ASIC algorithm over traditional and contemporary clustering methods through comparative analysis.

Evaluate the operational efficiency and clustering quality of ASIC using diverse datasets from real-world IoT applications.

Explore the scalability of ASIC and its effectiveness in handling real-time data adaptation without the need for reprocessing entire datasets.

Through these objectives, this study aims to provide a comprehensive understanding of incremental clustering's potential and establish a benchmark for future research in IoT data management.

2. Theoretical background and technical overview

The vast landscape of data processing, especially within the IoT, presents unique challenges and opportunities for advancing our analytical capabilities. The IoT generates a continuous stream of data that must be effectively managed and analyzed to derive actionable insights. This section outlines the fundamental methods and technologies employed in IoT data processing, along with an in-depth exploration of the advancements in incremental clustering analysis, highlighting its pivotal role in managing large-scale, dynamic data environments.

2.1 Basic methods and technologies for IoT data processing

In the IoT environment, data processing technologies are crucial for effective information management and decision support. The basic methods and technologies for IoT data processing encompass data collection, data preprocessing, data storage, and data analysis. These methods and technologies must efficiently handle large volumes of data and adapt to the dynamic changes of the data to ensure the timeliness and accuracy of data processing and analysis, which is vital for advancing the development and application of IoT technologies [4].

2.2 Progress and current state of incremental clustering analysis

Clustering analysis, a common unsupervised learning technique in the field of data mining, aims to group dataset samples based on similarity, ensuring high similarity within groups and low similarity between different groups [5].

Incremental clustering analysis, which adapts to the dynamic changes of data, is particularly suitable for scenarios where it is not feasible to process all data at once. Traditional batch processing clustering methods often encounter performance bottlenecks when facing large-scale data, whereas incremental clustering algorithms can update clustering results in real-time, making them more suitable for big data environments. This type of algorithm is especially apt for handling large-scale IoT data, which is typically generated in real-time and continues to grow in volume. Incremental clustering allows for effective management and analysis of these data, providing immediate insights and decision support.

A new incremental clustering algorithm has been proposed to optimize the clustering process within data streams. This algorithm updates the clustering structure in real-time, effectively managing new and disappearing data points and adapting to dynamic changes in data characteristics. Recent studies like those by Yang X. (2022) have introduced new incremental clustering algorithms optimized for big data characteristics. These studies have enhanced the precision and efficiency of algorithms like K-means by integrating Kalman filter techniques to adjust the settings of cluster centers [6]. Chefrour A. (2022) has improved the Incremental Density-Based Spatial Clustering of Applications with Noise

(IDBSCAN) by employing adaptive median filtering techniques, which better identify clusters of arbitrary shapes and recognize noise in dynamic databases, showcasing superior clustering quality and data processing speed compared to traditional methods [7]. Moreover, distributed incremental clustering algorithms have become a research focus in recent years. Inoubli et al. (2020) discussed the feasibility of applying incremental clustering algorithms on distributed platforms to handle data from various sources for global data analysis [8]. Their comprehensive review and analysis of the current state of distributed incremental clustering algorithms utilized bibliometric and word cloud analysis methods.

On another note, deep learning integrated methods have been introduced into incremental clustering algorithms to enhance the model's ability to recognize complex data patterns. These methods use neural networks to preprocess data, improving the traditional algorithms' limitations in handling high-dimensional data. Such enhancements not only improve the performance of clustering algorithms but also better address the noise and outliers encountered in practical applications. Additionally, researchers are focusing on the scalability and efficiency of these algorithms. For example, recent research has developed a graph-based incremental clustering algorithm that uses a dynamic graph model to manage relationships between data points and clusters, significantly enhancing the algorithm's performance in big data environments.

In application domains, incremental clustering analysis has shown unique advantages in sensitive topic detection, personalized medical analysis, and more. Zhang Y. (2015) proposed an incremental text clustering algorithm based on simhash for real-time internet news data, effectively enhancing the timeliness and accuracy of sensitive topic detection [9].

Incremental clustering methods have shown increased efficiency in environments with dynamically evolving data, such as real-time data visualization. These methods swiftly process new data points to generate instant cluster labels. Key studies include: Kushwah et al. (2021) reviewed various incremental clustering methods, emphasizing their efficiency in real-time visualization of large datasets [10].

In summary, research on incremental clustering analysis has made significant progress in recent years, with new algorithms and methods continuously being proposed to meet the growing data processing demands. Future research may achieve further breakthroughs in algorithm efficiency, processing capabilities, and optimizations for specific application scenarios, thereby better serving the increasing needs.

3. Incremental clustering method for IoT data

The continuous and voluminous data generation characteristic of IoT devices necessitates robust and dynamic data processing methods that can adapt in real-time to changing data characteristics. Incremental clustering, particularly suited for such environments, offers a promising solution. This section details the conceptual and practical frameworks of a novel incremental clustering method tailored for IoT data, ensuring scalability and responsiveness to the ever-evolving data landscapes typical of IoT ecosystems.

3.1 Method overview

Recent literature [11] introduces a novel online clustering algorithm called "Online Arbitrary Shaped Clustering through Correlated Gaussian Functions", which can generate clusters of arbitrary shapes in an unsupervised setting without relying on a predefined number of clusters. The algorithm determines clusters by analyzing the output of correlated Gaussian functions from the input data, a method considered more reliable than traditional backpropagation model optimization in a biological context. However, the algorithm initially processes only single data points at a time, which may lead to efficiency issues when handling large datasets [11]. To address this, we have optimized the algorithm by introducing a batch processing mechanism and developing an improved model called "Arbitrary Shaped Incremental Clustering" (ASIC) tailored for IoT data characteristics. The ASIC model, by batch processing data points, significantly enhances processing efficiency, making it more suitable for large-scale data environments and providing a practical clustering tool for IoT data analysis.

This adjustment allows the algorithm to efficiently process the entire dataset X, updating model parameters over multiple iterations to adapt to the clustering tasks of large-scale datasets. This involves extending the original scalar operations to vectorized or matrix operations to improve computational efficiency. Modifications were also made to the update rules, Gaussian function outputs, and any other operations directly dependent on individual data points.

When processing small batches XB of the entire dataset X, the formula for updating individual data points can be adjusted to updates based on batch data. This method not only improves computational efficiency but also reduces noise impact through batch gradient descent, potentially leading to more stable convergence performance. The following is applicable for small batch processing:

3.1.1 Batch calculation of Gaussian function outputs

Firstly, for a small batch XB of data points in the entire dataset X, calculate the output of each Gaussian function using the formula:

$$f_i(X_B) = \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{|B|}\left|x_j - \mu_{ij}\right|_2^2\right) \tag{1}$$

Where $|B|$ is the number of data points in the batch, $x_j$is the *j*-th feature of the data point, μ_{ij} is the center of the *i*-th Gaussian function on the *j*-th feature, and σ is the standard deviation.

3.1.2 Updating the Center of Gaussian Functions

For the small batch $X_B \subseteq X$, the update formula is adjusted to:

$$\Delta\mu_i = \frac{\eta}{\sigma|B|}\sum_{x\in X_B}\left(f_i(x)(x-\mu_i) - 2\lambda\sum_{j\neq i} f_i(\mu_j)(\mu_j-\mu_i)\right),$$

$$\mu_{i,new} = \mu_{i,old} - \eta\sum_{X_B}\Delta\mu_i \tag{2}$$

where η is the learning rate. This formula means that with each update, instead of considering the entire dataset X or individual data points, all data points in X_B are used to calculate $\Delta\mu_i$.

3.1.3 Updating the Scatter Matrix Q

Similarly, for the small batch X_B, the update formula for Q is adjusted to:

$$\Delta Q_{k,l} = \sum_{x\in X_B}\frac{f_k(x)f_l(x)}{|f_{1:K}(x)|_p^2} \tag{3}$$

3.1.4 Cumulative Updates

When cluster results are found to be unsatisfactory, indicating a concept drift, ΔQ should be accumulated into the existing Q matrix:

$$Q_{i,j} = Q_{i,j} + \Delta Q_{i,j} \tag{4}$$

3.1.5 Clustering Label Assignment

The assignment of clustering labels involves using the scatter matrix Q and correlation coefficients R to assign labels to Gaussian functions, determining which cluster each function belongs to. When processing the entire dataset, the method for calculating correlation coefficients R remains unchanged, but it is based on the updated scatter matrix Q of the entire dataset:

$$R_{k,l} = \frac{Q_{k,l}}{\sqrt{Q_{k,k}Q_{l,l}}} \tag{5}$$

For the entire dataset, the correlations of all Gaussian function outputs are calculated at once based on the correlation threshold τ to assign clustering labels.

3.2 Design of Incremental Clustering Algorithm

This paper proposes a framework for handling incremental clustering, segmenting the data and preserving the history of clustering centers to ensure that significant changes in each cluster's data points can be tracked. Clustering analysis is performed on the current block, and the clustering centers obtained are used with the next block of data for clustering analysis, ensuring that previous data information is not lost and maintaining data coherence. Parameters are also inherited from the last cluster update; if a significant deviation is found between the current and previous clustering center matrices, the clustering model is retrained to update parameters to ensure effective clustering results.

It should be noted that while this batch processing method can reduce update frequency and improve computational efficiency, it may also require adjustments to the learning rate $\eta\eta$ and other parameters to maintain the stability and effectiveness of the algorithm. In practical applications, this method may need to be appropriately adjusted and optimized based on specific problems and dataset characteristics.

• The rules for updating the Gaussian function centers μ_i and the scatter matrix Q need to be adjusted to handle the entire dataset X, not just individual data points.

• The outputs of the Gaussian functions need to be calculated through vectorized operations simultaneously for all data points in the dataset.

• The assignment of clustering labels depends on the updated scatter matrix Q and correlation coefficients R, which are calculated based on the results of processing the entire dataset. The batch processing method allows the algorithm to handle large datasets X more efficiently and stably. By extending operations that were originally aimed at single data points to handle a subset of the dataset at once in each iteration, it is possible to accelerate computation, reduce memory demands, and potentially help avoid local optima through the introduction of batch processing randomness.

Algorithm Description:

Algorithm Name: Arbitrary Distribution Incremental Clustering
Input: Stream data, number of cluster centers K, data dimensionality D, parameter σ, clustering center concentration degree λ, learning rate η;
Output: Cluster labels, cluster center results;

```
Initialize the model Model using Equation (1) with the specified number of cluster centers $K$, data dimensionality $D$, parameter $\sigma$, clustering center concentration degree $\lambda$, and learning rate $\alpha$.
Initialize the Clusterer using Equation (5) based on the model.
Set plotting limits and styles, and configure the data source.
Set the random seed and specify the batch size batch_count.

while True:
  for data points in data block <= batch_count:
    if not the first block:
      data block ← previous block's cluster centers
    endif
    data block ← fetch data
    batch_count += 1
  endfor
  Update the clusterer according to Equation (5)
  Obtain cluster centers and pseudo labels according to Equation (2)
  Plot the current state of clustering
  if drift occurs:
    Update the model according to Equations (3) and (4)
  endif
  batch_count ← batch_count + 1
endwhile
```

This pseudocode outlines a continuous process where the algorithm dynamically updates based on the data flow, adapting to changes in the data stream. Each iteration adjusts the clustering based on new data, allowing for incremental and continuous clustering suitable for large-scale dynamic datasets. This methodology ensures that the model remains current with evolving data trends and can handle real-time clustering challenges efficiently.

4. Experiment and results analysis

This section will detail the datasets utilized in the experiments, including their specific attributes which are critical for understanding the impact on the performance outcomes of the ASIC algorithm. Subsequent sections will delve into the methodologies for evaluating the clustering results, including the criteria and standards used to gauge the effectiveness and efficiency of the algorithm in various scenarios and data environments.

From the perspective of incremental clustering, the performance of the ASIC algorithm can be comprehensively assessed through metrics such as dataset size, runtime, and clustering evaluation indices.

4.1 Experimental datasets

In Table 1, we employ a variety of datasets to rigorously assess the adaptability and effectiveness of the ASIC algorithm across different scales and types of data. Our selection includes datasets with a broad spectrum of instances, dimensions, and classes, which are crucial for evaluating the robustness of the algorithm.

Table 1. Attributes of Data Sets Used in the Experiments

No.	Datasets	Classes	Dimensionality	Instances
1	AWS IoT [12]	5	6	6319485
2	Breast Cancer [13]	2	30	569
3	Digits [14]	10	64	1797
4	Iris [15]	3	4	150
5	Water Monitoring [16]	5	5	46893
6	Wind Turbine [17]	4	8	388151

The datasets utilized are:

AWS IoT: A large-scale dataset from the domain of the Internet of Things (IoT), comprising 6,319,485 instances with 6 dimensions across 5 distinct classes. This dataset is pivotal for testing the algorithm in high-volume, high-dimensional IoT scenarios.

Breast Cancer: A relatively small-scale dataset with 569 instances, featuring 30 dimensions and 2 classes. This medical dataset is essential for demonstrating the algorithm's capability in handling complex, high-dimensional medical data.

Digits: A medium-scale dataset that includes 1,797 instances with 64 dimensions across 10 classes. It represents a standard benchmark in image recognition tasks, testing the algorithm’s ability to process and classify high-dimensional data.

Iris: Another small-scale dataset, consisting of 150 instances with 4 dimensions across 3 classes. Despite its simplicity, this dataset remains a staple in pattern recognition and machine learning demonstrations.

Water Monitoring: An IoT dataset with 46,893 instances, 5 dimensions, and 5 classes. It serves an important role in examining the algorithm's performance in environmental data monitoring tasks within IoT applications.

Wind Turbine: This dataset contains 388,151 instances, spread across 8 dimensions and 4 classes. It focuses on IoT applications in renewable energy sectors, specifically wind energy, to evaluate the algorithm under different operational and environmental conditions.

The inclusion of IoT-specific datasets such as AWS IoT, Water Monitoring, and Wind Turbine underscores the growing importance of machine learning algorithms in managing and analyzing data from connected devices in real-time environments. These datasets, varying from small to large scale, enable a comprehensive examination of the ASIC algorithm's versatility in processing and analyzing diverse types of data, which is critical for its deployment in practical IoT systems. Through this diverse dataset portfolio, the study aims to establish a robust evaluation framework that will highlight the ASIC algorithm's strengths and potential areas for improvement in handling both conventional and IoT-specific challenges.

4.2 Evaluation Criteria

Cluster evaluation criteria are commonly used to measure the effectiveness of clustering algorithms, primarily including Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Silhouette Coefficient (SiC). These indicators have their own calculation formulas, suitable for evaluating different aspects of clustering performance.

If a dataset contains elements n (where n), with the true label set denoted as T and the clustering prediction result as Y, we can standardize the formulas of evaluation indicators as follows:

4.2.1 Adjusted Rand Index (ARI)

The Adjusted Rand Index (ARI) is a metric that measures the similarity between two clustering results, with higher values indicating greater similarity. ARI is the adjusted form of the Rand Index, designed to mitigate the influence of random clustering. The formula for ARI is as follows:

$$ARI = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Where TP represents the number of pairs of elements assigned to the same cluster in both T and Y, TN represents the number of pairs of elements assigned to different clusters in Y but the same cluster in T, FP represents the number of pairs of elements assigned to the same cluster in Y but different clusters in T, and FN represents the number of pairs of elements assigned to different clusters in both T and Y.

4.2.2 Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) is a metric based on information theory used to measure the similarity between two clustering results. Higher NMI values indicate greater consistency between clustering results. The formula for NMI is as follows:

$$NMI = \frac{I(T,Y)}{\sqrt{\mathrm{H}(T)\mathrm{H}(Y)}}$$

Where $(T$, $Y)$ represents the mutual information between the true label set T and the predicted label set Y, $H(T)$ represents the entropy of the true label set T, and $H(Y)$ represents the entropy of the predicted label set Y.

4.2.3 Silhouette Coefficient (SiC)

The Silhouette Coefficient (SiC) is a metric used to measure both the compactness and separation of clusters, with values ranging from [-1, 1]. A higher value indicates better clustering performance. The formula for SiC is as follows:

$$\mathrm{SiC} = \frac{1}{n}\sum_{i=1}^{n} \frac{b_i - a_i}{\max(a_i, b_i)}$$

Where, for each element x_i in the dataset:

- a_i: the average distance between the sample x_i and other samples in the same cluster in Y.
- b_i: the average distance between the sample $x_i$and samples in the nearest other cluster in Y.

4.3 Experimental Setup and Procedure

In this paper, the experimental setup was meticulously designed to provide a reliable and consistent framework for evaluating the performance of various clustering algorithms. The experiments were conducted using a robust hardware environment consisting of a computer equipped with an Intel-Core i9 CPU, 16 GB of RAM, and running the Windows 11 operating system. On the software side, Python 3.7, supported by Anaconda 3.8, formed the backbone of our experimental platform, ensuring compatibility and ease of implementation for various algorithmic modules.

For our comparative analysis, we selected three well-established clustering algorithms from the scikit-learn library: K-Means, Gaussian Mixture Models (GMM), and Agglomerative Clustering. These algorithms were chosen due to their wide acceptance and varied methodologies, providing a comprehensive baseline for assessing the proposed ASIC and OASC algorithms.

The configuration of the comparative algorithms was as follows:

• **K-Means and Agglomerative Clustering**: The number of clusters was set to correspond to the number of classes in each dataset to maintain consistency across methods. Except for this parameter, all other settings were left at their default values to ensure that the results were attributable to the intrinsic characteristics of each algorithm rather than external adjustments.

• **GMM**: This algorithm was also configured to use a number of components equal to the number of classes in the datasets, with other parameters remaining at their default settings. This approach allows for a fair comparison under similar constraints.

For the OASC and ASIC algorithms, we introduced specific parameter settings aimed at optimizing their performance relative to the structure of the datasets:

• **Number of Clusters**: Set to match the number of categories within each dataset.

• **Dimensionality (D)**: Considered explicitly in the parameter tuning to accommodate the specific characteristics of each dataset.

• **Sigma (σ)**: Fixed at 0.01 to control the sensitivity of the clustering process.

• **Clustering Center Aggregation Level (λ)**: Set at 0.5, balancing between fine granularity and broader clustering aggregations.

• **Learning Rate (η)**: Maintained at 0.01 to ensure stable convergence over iterations.

• **Data Points per Block**: For the ASIC algorithm, this was calculated as 10% of the total sample size, facilitating efficient data handling and processing during the clustering operations.

Each algorithm was executed ten times on the datasets listed in Table 1 to ensure statistical reliability. The results of these runs were carefully recorded and are presented in Figure 1 and Table 2, providing a transparent and detailed account of each algorithm’s performance across various data scenarios. This rigorous experimental procedure underscores our commitment to providing a thorough and unbiased evaluation of the clustering algorithms under study.

4.4 Experimental Results and Analysis

ASIC, as an incremental clustering algorithm, was designed to handle continuously growing and changing datasets, particularly suitable for large-scale real-time data processing.

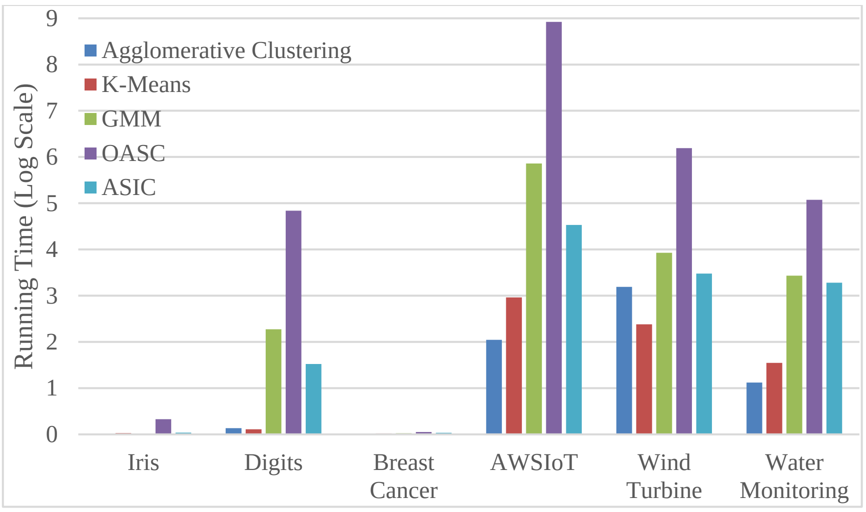

Figure 1. Comparison of Time Consumption of Different Algorithms on Datasets of Different Sizes

Figure 1 illustrates the runtime comparison between the ASIC algorithm and other clustering algorithms across datasets of varying sizes. On large-scale datasets such as AWS IoT, although ASIC's runtime is not the shortest, it exhibits a slight advantage over OASC and a significant improvement over K-Means and GMM. This is particularly crucial as incremental clustering algorithms ideally maintain lower runtime when processing large volumes of data, enabling frequent and real-time updates to clustering results to adapt to new data. Despite ASIC's longer runtime on large datasets, its performance remains within acceptable limits, especially in applications requiring dynamic updates to clustering. The design of the ASIC algorithm, by minimizing unnecessary computations and optimizing data processing, enables it to quickly adapt to data changes on large-scale datasets, effectively updating clustering results dynamically, showcasing its critical advantage in processing speed in big data environments.

Table 2 presents the cluster evaluation metrics (ARI, NMI, and SiC) for ASIC and other algorithms across different datasets. From the perspective of incremental clustering, the ASIC algorithm demonstrates superior performance in ARI and NMI metrics on most datasets. For instance, on the AWS IoT dataset, ASIC achieves an ARI of 0.6629, slightly lower than OASC's 0.6631 but higher than the results of K-Means and Agglomerative Clustering. In terms of the NMI metric, ASIC also exhibits high scores, particularly on the Water Monitoring dataset (0.6982), surpassing other algorithms. This result indicates ASIC's significant advantage in maintaining consistency between clustering results and true data labels.

Table 2. Average Cluster Evaluation Metrics for Each Algorithm Run 10 Times

Algorithm	Metric	Iris	Digits	Breast Cancer	AWSIoT	Wind Turbine	Water Monitoring
Agglomerative Clustering	ARI	0.7312	0.794	0.2872	0.6019	0.6801	0.4232
	NMI	0.7701	0.8682	0.3191	0.5801	0.7002	0.5393
	SiC	0.5543	0.1785	0.69	0.5887	0.6027	0.6779
K-Means	ARI	0.7302	0.6687	0.4914	0.5823	0.6491	0.6155
	NMI	0.7582	0.7469	0.4648	0.5921	0.7202	0.631
	SiC	0.5528	0.1825	0.6973	0.6461	0.5831	0.6839
GMM	ARI	0.9039	0.6778	0.8116	0.6628	0.6861	0.6381
	NMI	0.8997	0.7561	0.7061	0.6013	0.7399	0.6562
	SiC	0.5012	0.1774	0.5315	0.5891	0.6028	0.6552
OASC	ARI	0.9038	0.6789	0.8128	0.6631	0.687	0.6312
	NMI	0.8902	0.7563	0.7127	0.6041	0.7361	0.6841
	SiC	0.5007	0.1781	0.5319	0.5912	0.6001	0.6531
ASIC	ARI	0.9037	0.6819	0.8204	0.6629	0.6911	0.6444
	NMI	0.8931	0.7565	0.7119	0.6091	0.7282	0.6982
	SiC	0.5321	0.1626	0.5301	0.5927	0.5987	0.6499

Regarding the SiC metric, although ASIC's scores are slightly lower on some datasets, its overall performance remains robust, demonstrating its effectiveness in both intra-cluster compactness and inter-cluster separation, which are crucial for data clustering in dynamic environments. Considering its incremental clustering nature, this may be because the algorithm sacrifices some internal coherence when adapting to new data.

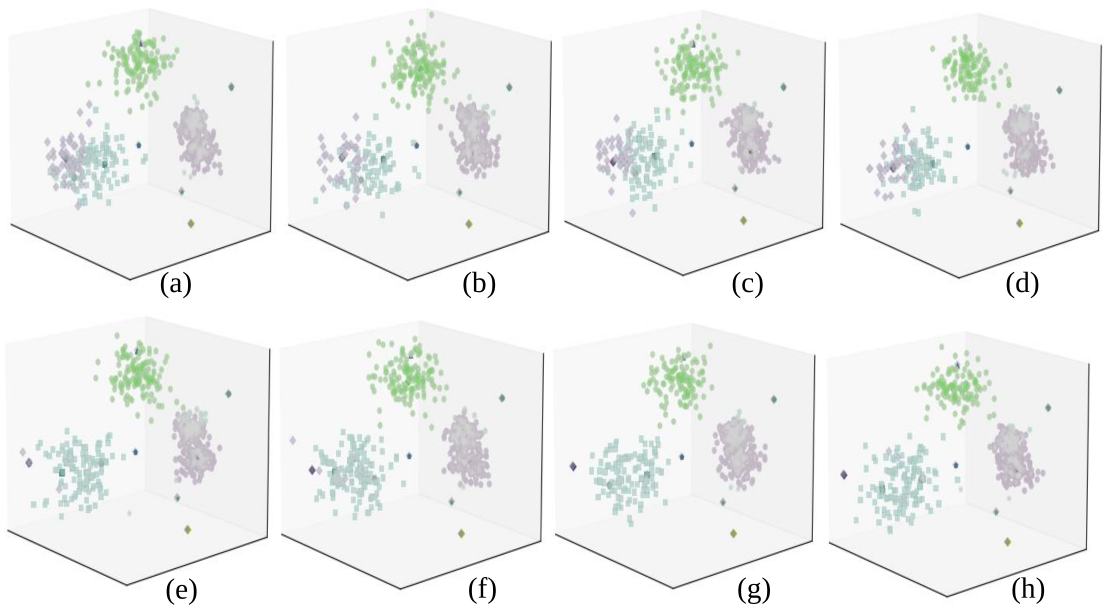

Figure 2. Presents the results of reducing the clustering outcome of the Arbitrary Distribution Incremental Clustering (ASIC) algorithm on AWS IoT data to three dimensions using the T-SNE algorithm

This dimensional reduction visualization allows for a comprehensive analysis of the ASIC algorithm's performance on the AWS IoT dataset. The figure demonstrates several distinct clusters represented by different colors in the three-dimensional space, showing the distribution of data points within the feature space. Ideally, points of the same color cluster together while distinctly separating from points of other colors, indicating the algorithm's capability to effectively identify and segregate different data subsets.

From figures (a) to (h), we observe well-separated clusters, indicating that the ASIC algorithm effectively distinguishes between different groups within IoT data. Additionally, the points within each cluster appear dense and compact, suggesting high intra-cluster similarity, aligning with the objectives of incremental clustering algorithms to ensure data points are proximate in feature space.

Furthermore, the shape and size of the clusters remain relatively consistent across different figures, indicating the stability of the ASIC algorithm. It shows consistent clustering results as new data continuously integrates, without being overly sensitive to new data, which is crucial in handling potentially noisy real-time IoT data streams.

Comparing figures (a) to (h), no anomalous clusters appear or disappear, suggesting that the ASIC algorithm does not suffer from overfitting or underfitting on this dataset. Instead, it demonstrates good adaptability and robustness necessary for maintaining cluster quality in non-static environments.

In conclusion, the performance of the ASIC algorithm on the AWS IoT dataset highlights its potent capability in handling large-scale and dynamically changing datasets. This algorithm not only demonstrates reliability in incremental clustering but also effectively manages the increase and variation in data volumes while maintaining cluster quality. Particularly, the three-dimensional visualization of its clustering results provides an intuitive means to assess algorithm performance, suitable for handling high-dimensional data typical in the IoT domain. While there are areas for improvement in certain performance metrics, the overall effectiveness and performance on small-scale datasets demonstrate the algorithm's broad applicability and flexibility across various real-time and large-scale data processing scenarios.

5. Conclusion and future prospects

As the integration of IoT within various sectors continues to expand, the demands for sophisticated data processing techniques that can keep pace with the rapid flow and transformation of data also increase. This paper introduces a robust solution in the form of the Arbitrary Distribution Incremental Clustering (ASIC) algorithm, specifically designed to address the dynamic and voluminous nature of IoT data streams. This section provides a comprehensive conclusion on the efficacy of the ASIC algorithm and outlines the potential directions for future research and advancements in this vital field.

This paper presents the Arbitrary Distribution Incremental Clustering (ASIC) algorithm tailored for IoT data, addressing the dynamic changes and real-time demands of IoT data effectively. Compared with traditional clustering algorithms and other incremental clustering methods, the ASIC algorithm exhibits superior clustering quality and operational efficiency in handling large-scale IoT datasets. Experimental results show that the ASIC algorithm significantly reduces computational resource consumption without sacrificing clustering accuracy, particularly excelling in environments with large data volumes and ongoing data distribution changes, showing its superior adaptability and flexibility.

By validating through multiple datasets of various scales and characteristics, the ASIC algorithm consistently performs well across all assessment metrics, especially in managing real-time changes in large-scale datasets like AWS IoT, quickly adapting to data changes, and effectively maintaining the stability of the clustering structure. Moreover, the algorithm responds swiftly to new data in the data stream, reflecting new patterns immediately, which is crucial for real-time data processing.

Although the ASIC algorithm has demonstrated strong performance, there is still room for further optimization. Future research could explore several directions: firstly, reducing the algorithm's time complexity and enhancing processing speed to better suit ultra-large-scale datasets. Secondly, enhancing the algorithm's robustness against noise and outliers to improve adaptability in complex data environments. Lastly, considering the integration of multiple data sources and heterogeneous data to boost the algorithm's application in multimodal data processing scenarios.

Through continuous research and improvement, incremental clustering algorithms are poised to play a more significant role in IoT and other real-time data processing fields, providing stronger support for data-driven decision-making.

Acknowledgements

This work was funded by the Fujian Provincial Education Research Project for Middle-aged and Young Teachers (Science and Technology category, No: JAT220739).

References

[1] Ullah, Inam, et al. "Data Science Meets Intelligent Internet of Things." Future Communication Systems Using Artificial Intelligence, Internet of Things and Data Science. 2024, CRC Press 73-91.
[2] Shen, Li, et al. "On efficient training of large-scale deep learning models: A literature review." arXiv preprint arXiv:2304.03589, 2023.

[3] Zhang, C., et al. "Deep Learning-Integrated Incremental Clustering for Streaming Data." IEEE Transactions on Neural Networks and Learning Systems, 2022.
[4] Naghib, Arezou, et al. "A comprehensive and systematic literature review on the big data management techniques in the internet of things." Wireless Networks, 29.3, 2023: 1085-1144.
[5] Oyewole, Gbeminiyi John, and George Alex Thopil. "Data clustering: application and trends." Artificial Intelligence Review, 56.7, 2023: 6439-6475.
[6] Yang, Xiaoqing. "Research on incremental clustering algorithm for big data." Applied Mathematics and Nonlinear Sciences, 8.2, 2023: 169-180.
[7] Chefrour, Aida. "A modified Incremental Density Based Clustering Algorithm." 2022 7th International Conference on Image and Signal Processing and their Applications (ISPA). IEEE, 2022.
[8] Inoubli, Wissem, et al. "A distributed and incremental algorithm for large-scale graph clustering." Future Generation Computer Systems, 134, 2022: 334-347.
[9] Zhang, Yuejin, Jiajia Zhang, and Dongmei Zhao. "Text clustering incremental algorithm in sensitive topic detection." Int J Inf Commun Sci, 3, 2018: 88.
[10] Yang, Xiaoqing. "Research on incremental clustering algorithm for big data." Applied Mathematics and Nonlinear Sciences, 8.2, 2023: 169-180.
[11] Eidheim, Ole Christian. "Online Arbitrary Shaped Clustering through Correlated Gaussian Functions." arXiv preprint arXiv:2302.06335, 2023.
[12] Aws-Samples. Aws-samples/aws-iotexamples. https://github.com/aws-samples/aws-iotexamples /tree/master/predictionDataSimulator, 2016.
[13] Dookeran, Keith A., et al. "Associations of two-pore domain potassium channels and triple negative breast cancer subtype in The Cancer Genome Atlas: systematic evaluation of gene expression and methylation." BMC research notes, 10, 2017: 1-9.
[14] Chen, Weijie, et al. "Self-supervised noisy label learning for source-free unsupervised domain adaptation." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2022.
[15] Pinto, Joylin Priya, Soumya Kelur, and Jyothi Shetty. "Iris flower species identification using machine learning approach." 2018 4th International Conference for Convergence in Technology (I2CT). IEEE, 2018.
[16] Kumar, Manish, et al. "Quality assessment and monitoring of river water using IoT infrastructure." IEEE Internet of Things Journal, 2023.
[17] Huang, Bin, et al. "Review of data-driven prognostics and health management techniques: lessions learned from PHM data challenge competitions." Machine Failure Prevention Technology, 2017, 1-17.