Toward Efficient and Incremental Spectral Clustering via Parametric Spectral Clustering

Jo-Chun Chen, Hung-Hsuan Chen
Department of Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
logchen0620@g.ncu.edu.tw, hhchen1105@acm.org

Abstract—Spectral clustering is a popular method for effectively clustering nonlinearly separable data. However, computational limitations, memory requirements, and the inability to perform incremental learning challenge its widespread application. To overcome these limitations, this paper introduces a novel approach called parametric spectral clustering (PSC). By extending the capabilities of spectral clustering, PSC addresses the challenges associated with big data and real-time scenarios and enables efficient incremental clustering with new data points. Experimental evaluations conducted on various open datasets demonstrate the superiority of PSC in terms of computational efficiency while achieving clustering quality mostly comparable to standard spectral clustering. The proposed approach has significant potential for incremental and real-time data analysis applications, facilitating timely and accurate clustering in dynamic and evolving datasets. The findings of this research contribute to the advancement of clustering techniques and open new avenues for efficient and effective data analysis. We publish the experimental code at https://github.com/109502518/PSC_BigData.

Index Terms—spectral clustering, incremental clustering, online clustering, nonlinear clustering

I. INTRODUCTION

Clustering, as a fundamental technique in various fields such as data mining and pattern recognition, plays a critical role in analyzing large datasets by dividing them into smaller and more coherent groups, facilitating the analysis of large datasets. The importance of clustering is further underscored by its frequent usage as a preprocessing step for subsequent analyses [1]. Despite the extensive development of clustering techniques, it remains an active research area due to its wide-ranging applications and inherent challenges [2].

Among the various clustering methods, spectral clustering has gained considerable popularity and demonstrated superior performance. Spectral clustering has a significant benefit in that it can manage datasets with patterns that are not linearly separable. This is accomplished by dividing the data points in a transformed space rather than directly dividing them in the original feature space. This unique property makes spectral clustering suitable for a variety of complex applications and domains, and it is often able to uncover clusters that other clustering techniques are unable to detect.

Although spectral clustering is a widely used clustering method that has shown remarkable performance in various applications, it suffers from significant limitations: its computation time and memory usage are huge, and it cannot perform

incremental clustering. Therefore, it could be challenging to apply spectral clustering to a large dataset. Furthermore, once a spectral clustering model is trained, it cannot cluster new data points without retraining the entire model. These limitations can be critical issues in applications with big data or when the data streams continuously arrive, and the clustering results must be updated in real-time.

To overcome these limitations and enable spectral clustering for handling big data and supporting incremental clustering, this paper presents an innovative approach called parametric spectral clustering (PSC). Essentially, PSC learns a parametric function to project data points into low-dimensional representations. This is a critical step in spectral clustering, although it requires a lot of computing resources. The approximated low-dimensional representations of both the new and original data points are then utilized altogether for clustering.

We evaluate the effectiveness and efficiency of PSC based on real-world datasets. The experimental results demonstrate that PSC handles big data more efficiently and enables incremental clustering while maintaining clustering quality comparable to the standard spectral clustering method. Thus, PSC has broad applications in various fields with large datasets, such as user behavior segmentation, social media content grouping, real-time anomaly detection, and many more [3]–[6].

The rest of the paper is organized as follows. Section II reviews previous studies on spectral clustering and incremental clustering. Section III introduces the training process, the inference process, and the properties of PSC. Section IV presents the experimental results. Finally, Section V summarizes our contributions and discusses future research directions.

II. RELATED WORK

Spectral clustering leverages the eigenvalues and eigenvectors of a Laplacian matrix to transform the data into a lower-dimensional but more representative space, where a base clustering algorithm, such as k-means, can be applied to the transformed data instances. Spectral clustering has been extensively discussed in the literature, with notable works including [7], [8]. Numerous variants and enhancements of spectral clustering have been proposed to improve its robustness and scalability [9]–[12].

Incremental clustering (a.k.a. online clustering) addresses the challenge of clustering large datasets in real-time. Unlike

Algorithm 1 Spectral clustering algorithm
***

Input: $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$: n data points, each $\boldsymbol{x}_i \in R^d$

Input: p: the length of the low-dimensional space, $p < d$

Output: $\boldsymbol{C} = \{c_1, \ldots, c_n\}$: the cluster IDs for the input data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$

1: Compute the similarity matrix $\boldsymbol{S} = [s_{i,j}] \in R^{n \times n}$, where $s_{i,j}$ represents the similarity score between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

2: Compute the normalized Laplacian matrix $\boldsymbol{L} = \boldsymbol{D}^{-1/2} \boldsymbol{S} \boldsymbol{D}^{-1/2}$, where $\boldsymbol{D} = [d_{i,j}] \in R^{n \times n}$ is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{n} s_{i,j}$

3: Compute the p eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ of $\boldsymbol{L}$ whose corresponding eigenvalues are the p largest ones and form the matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p] \in R^{n \times p}$ (i.e., the columns of $\boldsymbol{V}$ are the eigenvectors)

4: Treat each row in $\boldsymbol{V}$ as a data point in R^p and apply a clustering algorithm (e.g., k-means) on these data points. The output cluster IDs $c_1, \ldots c_n$ are that of the data points $\boldsymbol{x}_1, \ldots \boldsymbol{x}_n$.
***

traditional batch clustering, incremental clustering algorithms continuously cluster new data points (and possibly update the cluster ID of the original data points) as the new data points arrive. This incremental property allows efficient handling of dynamic datasets and enables the detection of evolving patterns. Various approaches have been proposed for incremental clustering, including stream clustering algorithms [13], [14] and online clustering methods [15]. Famous algorithms alone this line includes CluStream [16], DenStream [17], and BIRCH [18]. However, these models are mostly used in the data stream environment and are sometimes sensitive to the selected hyperparameters. Additionally, studies on data streams commonly assume that old data points become less important, which may ease the computation burden and accelerate the computation. However, gradually retiring old data points may not be a valid assumption in many cases. A tutorial on online clustering can be found in [19].

Previous studies have addressed the computational challenges associated with spectral clustering, primarily through approximation or sampling techniques [20], [21]. In contrast to these methods, our proposed method is distinct by leveraging the parametric modeling technique, making PSC stand out in comparison to prior work.

III. METHODOLOGY

A. Preliminary: Spectral Clustering

Spectral clustering transforms input data points into a low-dimensional (but probably more representative) space and clusters the data points in the low-dimensional space. The transformation uses spectral properties, often identifying distinctive patterns not easily detected by alternative clustering algorithms.

The spectral clustering algorithm separates data points according to several key steps [7]. The algorithm begins by computing a similarity matrix (typically using the Gaussian

TABLE I
THE STRUCTURE OF A SIMPLE NEURAL NETWORK THAT LEARNS TO MAP $\boldsymbol{x}_i \in R^d$ TO A SMALLER VECTOR $\boldsymbol{v}_{i*} \in R^p$ ($p < d$).

Layer Type	Number of Neurons	Activation Function
Input feature	d	None
First fully connected layer	n_1	ReLU
Second fully connected layer	n_2	ReLU
Third fully connected layer	n_3	ReLU
Fourth fully connected layer	p	None

kernel as a proxy of the similarity score, which will be explained in Section III-B) that captures the pairwise distances between the input data points. This similarity matrix is the foundation for constructing a graph representation of the dataset, with the data points acting as nodes, and the edges and weights reflecting the pairwise similarities. The algorithm calculates the Laplacian matrix from this graph representation that captures the relationships among the data points. The next step involves calculating the eigenvalues and eigenvectors of the Laplacian matrix. Finally, these eigenvectors serve as the transformed lower-dimensional representation, where a base clustering algorithm, such as k-means, can be applied. Algorithm 1 provides the pseudocode.

Although spectral clustering is powerful, it requires many computational resources in terms of time and space. First, the computational demands associated with spectral clustering are intensive. Tasks such as eigenvalue decomposition (as seen in line 3 of Algorithm 1) and matrix operations (as seen in line 2 of Algorithm 1) are computationally expensive and typically exhibit a time complexity of $O(n^3)$. Furthermore, the memory requirements for storing the similarity matrix and performing eigenvalue decomposition can quickly become prohibitive as the size of the dataset increases with the complexity of the space of $O(n^2)$ [21]. These computational and memory constraints limit the scalability of spectral clustering.

Another limitation of spectral clustering is the lack of support for incremental clustering. Once the spectral clustering model is trained on a fixed dataset, incorporating new data points, even just a single data point, into the existing model without retraining the entire model becomes challenging. This limitation hinders the application of spectral clustering in scenarios involving enormous datasets clustering or data stream clustering, as it becomes impractical to retrain the model whenever new data points arrive in real-time.

B. Parametric Spectral Clustering – Training

Parametric spectral clustering consists of two key steps. The first step is almost identical to the first half of the spectral clustering algorithm. Given a sampling rate r ($0 < r \leq 1$), we sample $\nu = \text{round}(nr)$ data points $\boldsymbol{X}' = \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_\nu\}$ from the initially available data points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. We treat each data point $\boldsymbol{x}'_i \in R^d$ as a node and connect all edges in the graph. Next, we select a kernel function (typically a Gaussian kernel [7]) to define $s_{i,j}$ the similarity between two data points $\boldsymbol{x}'_i$ and $\boldsymbol{x}'_j$. Once the similarity matrix $\boldsymbol{S} = [s_{i,j}] \in R^{\nu \times \nu}$ is

Algorithm 2 Training procedure of PSC

Input: $X = \{x_1, \ldots, x_n\}$: n data points, each $x_i \in R^d$
Input: p: the length of the low-dimensional space, $p < d$
Input: r: sampling rate, $0 < r \le 1$
Output: A trained model M
1: $\nu = \text{round}(rn)$
2: Sample ν data points $X' = \{x'_1, \ldots, x'_\nu\}$ from X
3: Compute the similarity matrix $S = [s_{i,j}] \in R^{\nu \times \nu}$, where $s_{i,j}$ represents the similarity score between x'_i and x'_j (usually using Gaussian kernel)
4: Compute the normalized Laplacian matrix $L = D^{-1/2} S D^{-1/2}$, where $D = [d_{i,j}] \in R^{\nu \times \nu}$ is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{n} s_{i,j}$
5: Compute the p eigenvectors $v_1, \ldots, v_p$ of L whose corresponding eigenvalues are the p largest ones and form the matrix $V = [v_1, \ldots, v_p] \in R^{\nu \times p}$, i.e., with the eigenvectors as columns of V
6: Train a neural network M (structure in Table I) that maps x_i to v_{i*} (row i of matrix V)

Algorithm 3 Inference (clustering) procedure of PSC

Input: $X = \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}\}$: $n + m$ data points. The first n data points are available initially; the extra m data points need to be clustered but available after model training
Input: A trained model M
Output: $C = \{c_1, \ldots, c_n, c_{n+1}, \ldots, c_{n+m}\}$: the cluste IDs of $n + m$ data points in X
1: **for** $i = 1$ to $n + m$ **do**
2: $\quad u_i \leftarrow M(x_i)$
3: Apply a base clustering algorithm (e.g., k-means) to cluster data points $u_1, \ldots, u_{n+m}$. The output cluster IDs $c_1, \ldots, c_{n+m}$ for $u_1, \ldots, u_{n+m}$ is that of the data point for $x_1, \ldots, x_{n+m}$

---

generated, we construct the Laplacian matrix L and compute $v_1, \ldots, v_p$ the p eigenvectors of L whose eigenvalues are the largest. We create a matrix $V = [v_{i,j}] \in R^{\nu \times p}$ where the ith row in matrix V is denoted as v_{i*}. This matrix V is a crucial component in the subsequent training phase.

The second step aims to train a supervised learning model M that maps a training instance x_i to v_{i*}, the ith row of matrix V, in which the columns are the p largest eigenvectors of the Laplacian matrix L. In other words, the model M learns to project a data instance onto the directions of the p most important principal components. We apply a simple multilayer perceptron (MLP) for the model M, whose structure is illustrated in Table I. However, it can be easily replaced with other neural networks or supervised learning models.

The training procedure for the parametric spectral clustering algorithm is given in Algorithm 2.

C. Parametric Spectral Clustering – Inference

Regarding the inference or clustering phase in PSC, we aim to efficiently cluster an instance x_i, even if x_i does not appear in the initial training data. The model M obtained in the training phase converts each x_i into u_i, which serves as an approximation of the p most significant representations of x_i. Using this transformation, we can efficiently emulate the computations performed in lines 1 to 4 of Algorithm 1, eliminating the need to construct large similarity and Laplacian matrices and perform computationally expensive matrix operations and eigendecomposition. Consequently, we achieve significant computational savings in time and space but still generate low-dimensional embeddings for the data points x_i-s that well preserve the main characteristics.

Once each instance x_i has been transformed into u_i, we apply a base clustering algorithm, e.g., k-means, on u_i-s. The resulting cluster assignments for u_i-s are then considered cluster assignments for x_i-s. We follow the convention by employing the popular k-means algorithm as the clustering method for the low-dimensional data points. However, other base clustering algorithms, e.g., DBSCAN and the Gaussian Mixture Model, can also be applied.

Algorithm 3 gives the inference/clustering procedure.

IV. EXPERIMENTS

A. Experimental datasets

We conducted experiments on both tabular datasets and image datasets. The tabular datasets include the famous Iris, Wine, and BreastCancer datasets. The image datasets include UCIHW, MNIST, and Fashion-MNIST.

B. A Comparison of Clustering Quality

We would like to know whether PSC's clustering result is comparable to SC. We compare the SC and PSC's clustering results based on clustering accuracy (ClusterAcc), adjusted rand index (ARI), and adjusted mutual information (AMI).

Since the experimental datasets contain ground-truth labels, we use these labels to evaluate the quality of a clustering algorithm. However, a clustering algorithm may assign an arbitrary label to each cluster it forms. Therefore, the standard accuracy computation usually underestimates the clustering result. For example, given 5 instances with ground-truth labels as $[1, 1, 2, 2, 3]$, if a clustering algorithm assigns their cluster IDs as $[2, 2, 3, 3, 1]$, the standard accuracy computation outputs 0% because none of the predicted cluster IDs matches the ground-truth labels. However, a careful reexamination reveals that the clustering is perfect because all instances of the same ground-truth class belong to the same predicted cluster (and vice versa). Essentially, if two labeled lists are given, and one can become another simply by remapping the label IDs, the cluster assignments of the two lists are identical. In the above example, remapping labels 2 to 1, 3 to 2, and 1 to 3 for the second list makes the two lists identical. As a result, we define the clustering accuracy based on the equation below.

TABLE II
A COMPARISON OF THE CLUSTERING QUALITY OF SPECTRAL CLUSTERING AND PARAMETRIC SPECTRAL CLUSTERING. WE REPORT THE MEAN $\pm$ STANDARD DEVIATION FOR EACH SCORE.

	SC			PSC		
	ClusterAcc	ARI	AMI	ClusterAcc	ARI	AMI
Iris	0.889 ± 0.00	0.712 ± 0.00	0.77 ± 0.00	0.92 ± 0.025	0.781 ± 0.06	0.813 ± 0.038
Wine	0.963 ± 0.00	0.876 ± 0.00	0.855 ± 0.00	0.966 ± 0.02	0.892 ± 0.061	0.875 ± 0.071
BreastCancer	0.953 ± 0.00	0.819 ± 0.00	0.732 ± 0.00	0.932 ± 0.018	0.745 ± 0.062	0.632 ± 0.071
UCIHW	0.801 ± 0.067	0.743 ± 0.078	0.858 ± 0.026	0.825 ± 0.061	0.764 ± 0.055	0.855 ± 0.026
MNIST	0.794 ± 0.04	0.748 ± 0.03	0.842 ± 0.013	0.775 ± 0.046	0.73 ± 0.049	0.839 ± 0.013
Fashion-MNIST	0.609 ± 0.015	0.484 ± 0.009	0.644 ± 0.009	0.615 ± 0.041	0.475 ± 0.028	0.634 ± 0.009

TABLE III
THE NETWORK STRUCTURE OF THE AUTOENCODER TO CONVERT IMAGES IN MNIST INTO EMBEDDINGS (w: WIDTH OF INPUT IMAGE; h: HEIGHT OF INPUT IMAGE).

Layer	Type	Input shape	Output shape	Activation
1	Flatten	(w, h)	$w \times h$	ReLU
2	Linear	$w \times h$	1568	ReLU
3	Linear	1568	784	ReLU
4	Linear	784	392	ReLU
5	Linear	392	49	None
6	Linear	49	392	ReLU
7	Linear	392	784	ReLU
8	Linear	784	1568	ReLU
9	Linear	1568	$w \times h$	None
10	Unflatten	$w \times h$	(w, h)	None

$$ClusterAcc\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) := \max_{\forall \check{\boldsymbol{y}} \in P(\hat{\boldsymbol{y}})} \left\{ \frac{1}{n} \sum_{i=1}^{n} I\left(\check{y}^{(i)} = y^{(i)} \right) \right\}, \tag{1}$$

where $\boldsymbol{y} = [y^{(1)}, \dots, y^{(n)}]$ is the list of ground-truth labels of size n, $\hat{\boldsymbol{y}} = [\hat{y}^{(1)}, \dots, \hat{y}^{(n)}]$ is a list of predicted cluster IDs, $P(\hat{\boldsymbol{y}})$ returns a set of all identical lists for $\hat{\boldsymbol{y}}$, $I()$ is an indicator function, and $\check{\boldsymbol{y}} = [\check{y}^{(1)}, \dots, \check{y}^{(n)}]$ is an identical list of $\hat{\boldsymbol{y}}$. In other words, we compute the accuracies based on the ground-truth label list with all the identical lists of the prediction and return the largest accuracy score.

In addition to clustering accuracy, we also use ARI and AMI to evaluate the quality of the clustering results. ARI and AMI are biased towards different clustering assignments: ARI likes a balanced partition [22], and AMI favors an unbalanced partition. We report both to fairly assess different situations.

Since both SC and PSC are nondeterministic, executing either multiple times on a consistent dataset produces varied clustering outcomes. To fairly evaluate each method, we conduct each experiment 5 times and subsequently present the mean and standard deviation derived from these 5 trials.

Table II gives the three quality measurements for the SC and PSC clustering results in the six datasets. SC and PSC yield comparable results on all datasets. Interestingly, PSC sometimes outperforms SC even though PSC approximates the low-dimensional representations computed by SC. This is probably because the learning process implicitly introduces regularization factors so that PSC learns a better mapping.

Raw images are often characterized by complex structures, which hinders the effectiveness of traditional clustering algorithms. Therefore, we applied an autoencoder as a preprocessing step to extract the essential information of each image in the MNIST dataset into a compact vector that captures the critical patterns in an image. The structure of the autoencoder is shown in Table III. We use the bottleneck layer (the output of layer 5 in Table III) as the compact representation of the input image. Fashion-MNIST images are more complicated than MNIST, so we use an image autoencoder to convert images into vectors so that the spatial structures are reserved. Table IV gives the structure of the image autoencoder. We use the output of layer 7 as the extracted features.

Regarding the hyperparameters, we define n_1, n_2, n_3 (as specified in Table I) as follows: 32, 64, 32 for the Iris dataset; 26, 52, 26 for the Wine dataset; 60, 120, 60 for the BreastCancer dataset; 196, 392, 196 for both the MNIST dataset and the Fashion-MNIST. While further fine-tuning these hyperparameters may yield better clustering results, we did not spend much effort tuning them, as our primary focus is to show that PSC can cluster big data.

C. Training Data Size, Clustering Quality, Empirical Execution Time, and Empirical Memory Usage

In Section IV-B, we have demonstrated the quality of SC and PSC clustering using identical training instances. However, because PSC utilizes a parametric model to map data points to lower-dimensional representations, there exists an opportunity to use a subset of the provided data points as training data for PSC. Here, we investigate the efficiency and effectiveness of PSC utilizing a subset of training instances.

In particular, this section compares execution time, peak memory consumption, and clustering quality between SC and PSC, considering the varying rates of training data sampling for PSC. We use the MNIST dataset in this section for experiments. Note that applying spectral clustering on the MNIST dataset using a modern laptop is sometimes infeasible due to the substantial memory requirements of the modern spectral clustering algorithm. For example, if we want to cluster the entire MNIST dataset (which contains $70,000$ samples) using spectral clustering, a naïve implementation requires creating large matrices (e.g., $\boldsymbol{D}$ and $\boldsymbol{L}$ in Algorithm 1), each of them with shape $70,000 \times 70,000$. If each entry is a

TABLE IV

THE CONVOLUTIONAL NEURAL NETWORK STRUCTURE OF THE AUTOENCODER TO CONVERT IMAGES IN FASHION-MNIST INTO EMBEDDINGS.

Layer	Type	Input shape	Output shape	Kernel num	Kernel size	Stride	Padding	with BN	Activation
1	Conv.	$(w, h, 1)$	$(w, h, 16)$	16	5×5	1	Same	True	ReLU
2	Pooling	$(w, h, 16)$	$(w/2, h/2, 16)$	None	2×2	2	Valid	False	None
3	Conv.	$(w/2, h/2, 16)$	$(w/2, h/2, 32)$	32	3×3	1	Same	True	ReLU
4	Pooling	$(w/2, h/2, 32)$	$(w/4, h/4, 32)$	None	2×2	2	Valid	False	None
5	Conv.	$(w/4, h/4, 32)$	$(w/4, h/4, 1)$	1	3×3	1	Same	False	Sigmoid
6	ConvTranspose.	$(w/4, h/4, 1)$	$(w/4, h/4, 32)$	32	3×3	1	Same	True	ReLU
7	ConvTranspose.	$(w/4, h/4, 32)$	$(w/2, h/2, 16)$	16	3×3	2	Double	True	ReLU
8	ConvTranspose.	$(w/2, h/2, 16)$	$(w, h, 1)$	1	5×5	2	Double	False	Sigmoid

TABLE V

A COMPARISON OF SC AND PSC WITH DIFFERENT SAMPLING RATIOS IN TERMS OF THE EXECUTION TIME (SECONDS), PEAK MEMORY USAGE (MB), AND CLUSTERING QUALITY ON THE MNIST DATASET.

Method	Execution time (s)	Peak memory usage (MB)	ClusterAcc	ARI	AMI
SC	2462	5331	0.794 ± 0.04	0.748 ± 0.03	0.842 ± 0.014
PSC ($r = 1/6$)	Training: 453 ($\downarrow 82\%$) Inference: 0.443 ± 0.027	Training: 1032 ($\downarrow 81\%$) Inference: 92.83 ± 1.965	0.732 ± 0.076	0.684 ± 0.068	0.792 ± 0.03
PSC ($r = 2/6$)	Training: 533 ($\downarrow 78\%$) Inference: 0.419 ± 0.088	Training: 1717 ($\downarrow 68\%$) Inference: 92.63 ± 2.09	0.739 ± 0.089	0.68 ± 0.066	0.798 ± 0.02
PSC ($r = 3/6$)	Training: 706 ($\downarrow 71\%$) Inference: 0.437 ± 0.048	Training: 2532 ($\downarrow 53\%$) Inference: 96.21 ± 1.785	0.764 ± 0.041	0.712 ± 0.044	0.825 ± 0.016
PSC ($r = 4/6$)	Training: 1029 ($\downarrow 58\%$) Inference: 0.379 ± 0.06	Training: 3328 ($\downarrow 38\%$) Inference: 96.54 ± 2.271	0.775 ± 0.046	0.73 ± 0.049	0.839 ± 0.013
PSC ($r = 5/6$)	Training: 1472 ($\downarrow 40\%$) Inference: 0.491 ± 0.036	Training: 4937 ($\downarrow 7\%$) Inference: 96.5 ± 2.15	0.819 ± 0.039	0.753 ± 0.023	0.84 ± 0.01

single-precision floating point (4 bytes), a single matrix needs $4 \text{ bytes} \times 70,000^2 \approx 19.6$ GB.

We empirically explore the influence of different training data sampling rates on the clustering quality, execution time, and memory utilization of parametric spectral clustering when applied to the MNIST dataset. Table V describes the experimental results. First, the training and inference duration of the PSC is considerably faster than the execution time of the SC. The PSC's training process requires only a portion of the training data, which is why it is faster than SC. The inference time for PSC is almost negligible. Second, for the same reason, the peak memory utilization during the PSC's training and inference stages remains below the SC's peak memory requirement. The peak memory consumption is approximated using the memory profiler `mprof`.[1] Third, we study the connection between the training data sampling rate and the clustering quality achieved by the PSC. The results demonstrate that increasing the sampling rate can lead to an improvement in clustering quality. However, the enhancement is not considerable, and the gap becomes insignificant when the sampling rate exceeds a certain threshold. In essence, after reaching this threshold, the incremental benefit arising from including additional training instances becomes negligible and can significantly increase the cost of training. Consequently, we conclude that the PSC generates satisfactory clustering results

using more economical training resources without sacrificing substantial clustering quality.

D. Quality and Computational Cost of Incremental Clustering

This section demonstrates the efficacy of PSC in handling incremental clustering tasks.

We use a subset of 20,000 instances from MNIST as initial training data to train the PSC model. Subsequently, we simulate an incremental clustering scenario by introducing incremental batches of new instances that require clustering. Specifically, we consider scenarios with additional samples of sizes 2,000, 4,000, 6,000, 8,000, and 10,000. PSC is requested to cluster the initial 20,000 instances and the extra samples using the model trained only on the initial 20,000 training instances, thus fulfilling the incremental clustering.

The results of these incremental clustering simulations are presented in Table VI. As the number of newly arrived instances increases, there are slight changes in both the clustering time and the peak memory usage. This result validates that the PSC exhibits remarkable efficiency for incremental clustering tasks. Meanwhile, including more extra instances seems to improve the clustering quality marginally. We suspect that more instances may reveal a clearer pattern among different groups, thereby increasing the quality of the clustering results. On the contrary, SC is not able to manage incremental clustering. Therefore, even adding a small number of data points requires complete retraining, which is time consuming and memory intensive.

[1] https://pypi.org/project/memory-profiler/

TABLE VI
THE QUALITY AND COMPUTATION COST OF INCREMENTAL CLUSTERING WHEN THE NUMBER OF EXTRA INSTANCES VARIES

Size of extra instances	Clustering time (s)	Peak memory usage (MB)	ClusterAcc	ARI	AMI
2,000	0.569 ± 0.012	239.8 ± 2.949	0.738 ± 0.045	0.675 ± 0.054	0.811 ± 0.022
4,000	0.59 ± 0.026	198.6 ± 3.66	0.74 ± 0.042	0.705 ± 0.022	0.82 ± 0.01
6,000	0.634 ± 0.022	207.9 ± 3.145	0.745 ± 0.071	0.673 ± 0.099	0.816 ± 0.032
8,000	0.681 ± 0.095	262.4 ± 4.366	0.744 ± 0.052	0.705 ± 0.039	0.826 ± 0.018
10,000	0.659 ± 0.058	272.7 ± 3.838	0.795 ± 0.057	0.739 ± 0.072	0.844 ± 0.026

V. DISCUSSION

In this paper, we have proposed an innovative approach called parametric spectral clustering to address the limitations of traditional spectral clustering in the handling of large datasets and the support of incremental clustering. By leveraging a combination of training with a subset of instances and efficient low-dimensional projection, the PSC achieves efficient computation and economical memory usage while maintaining competitive clustering accuracy. Furthermore, we have demonstrated the effectiveness of PSC through extensive experiments on various datasets. Our results show that the PSC outperforms traditional spectral clustering regarding computational efficiency and peak memory usage. Additionally, we have validated the applicability of PSC in online clustering scenarios, where new data points can be clustered on the fly without retraining the entire model. Overall, our contribution extends the application of spectral clustering to domains with big data and real-time clustering requirements, thus opening up new possibilities for efficient and effective nonlinear clustering in various fields.

As an ongoing work, we are developing a user-friendly library that provides a scikit-learn-like interface for PSC. Users can use PSC with the commands `psc.fit(X)` and `psc.predict(X)` to cluster the data points. This library will simplify the implementation and usage of PSC, offering seamless integration with existing machine learning workflows. By providing accessible tools and resources, we aim to empower a broader range of users, including practitioners and researchers with varying levels of expertise in machine learning and data analysis, to explore the potential of PSC in their projects. This will facilitate greater adoption and further advances in the field, fostering collaboration and knowledge exchange among the research community.

A potential issue with the PSC is its inefficiency in adapting to data drift. The computation of a new instance's ground-truth embedding through eigendecomposition in spectral clustering makes the acquisition of ground-truth labels for training targets resource-intensive. Therefore, developing methods to efficiently manage data drift is a direction we are eager to explore.

ACKNOWLEDGEMENT

REFERENCES

[1] C.-Y. Lien, G.-J. Bai, and H.-H. Chen, "Visited websites may reveal users' demographic information and personality," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019, pp. 248–252.

[2] J. Kleinberg, "An impossibility theorem for clustering," *Advances in neural information processing systems*, pp. 463–470, 2003.

[3] R.-Y. Wang and H.-H. Chen, "Detecting inactive cyberwarriors from online forums," in *IEEE International Conference on Web Intelligence*, 2023.

[4] G.-J. Bai, C.-Y. Lien, and H.-H. Chen, "Co-learning multiple browsing tendencies of a user by matrix factorization-based multitask learning," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019, pp. 253–257.

[5] D. Y. Wu, T.-H. Lin, X.-R. Zhang, C.-P. Chen, J.-H. Chen, and H.-H. Chen, "Detecting inaccurate sensors on a large-scale sensor network using centralized and localized graph neural networks," *IEEE Sensors Journal*, vol. 23, no. 15, pp. 16 446–16 455, 2023.

[6] T.-H. Lin, X.-R. Zhang, C.-P. Chen, J.-H. Chen, and H.-H. Chen, "Learning to identify malfunctioning sensors in a large-scale sensor network," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 2582–2590, 2021.

[7] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[8] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[9] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[10] X. Zhu, S. Zhang, Y. Li, J. Zhang, L. Yang, and Y. Fang, "Low-rank sparse subspace for spectral clustering," *IEEE Transactions on knowledge and data engineering*, vol. 31, no. 8, pp. 1532–1543, 2018.

[11] R. Couillet and F. Benaych-Georges, "Kernel spectral clustering of large dimensional data," 2016.

[12] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 568–586, 2010.

[13] S. Ding, F. Wu, J. Qian, H. Jia, and F. Jin, "Research on data stream clustering algorithms," *Artificial Intelligence Review*, vol. 43, pp. 593–600, 2015.

[14] C. C. Aggarwal, "A survey of stream clustering algorithms," in *Data Clustering*. Chapman and Hall/CRC, 2018, pp. 231–258.

[15] W. Barbakh and C. Fyfe, "Online clustering algorithms," *International journal of neural systems*, vol. 18, no. 03, pp. 185–194, 2008.

[16] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*. Elsevier, 2003, pp. 81–92.

[17] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[19] J. Montiel, H.-A. Ngo, M.-H. Le-Nguyen, and A. Bifet, "Online clustering: Algorithms, evaluation, metrics, applications and benchmarking," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4808–4809.

[20] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 907–916.

[21] M. Li, X.-C. Lian, J. T. Kwok, and B.-L. Lu, "Time and space efficient spectral clustering via column sampling," in *CVPR 2011*. IEEE, 2011, pp. 2297–2304.

[22] S. Romano, N. X. Vinh, J. Bailey, and K. Verspoor, "Adjusting for chance clustering comparison measures," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 4635–4666, 2016.