Array 15 (2022) 100192

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

A parameter-free graph reduction for spectral clustering and SpectralNet

Mashaan Alshammari [a,*], John Stavrakakis [b], Masahiro Takatsuka [b]

[a] College of Computer Sciences and Engineering, University of Hail, Hail 81411, Saudi Arabia
[b] School of Computer Science, The University of Sydney, NSW 2006, Australia

ARTICLE INFO

Keywords:
Spectral clustering
SpectralNet
Graph reduction
Local scale similarity

ABSTRACT

Graph-based clustering methods like spectral clustering and SpectralNet are very efficient in detecting clusters of non-convex shapes. Unlike the popular k-means, graph-based clustering methods do not assume that each cluster has a single mean. However, these methods need a graph where vertices in the same cluster are connected by edges of large weights. To achieve this goal, many studies have proposed graph reduction methods with parameters. Unfortunately, these parameters have to be tuned for every dataset. We introduce a graph reduction method that does not require any parameters. First, the distances from every point p to its neighbors are filtered using an adaptive threshold to only keep neighbors with similar surrounding density. Second, the similarities with close neighbors are computed and only high similarities are kept. The edges that survive these two filtering steps form the constructed graph that was passed to spectral clustering and SpectralNet. The experiments showed that our method provides a stable alternative, where other methods' performance fluctuated according to the setting of their parameters.

1. Introduction

The problem of detecting clusters of non-convex geometric shape, has been long studied in the literature of pattern recognition. The solutions of this problem could be broadly classified into two categories: kernel- and graph-based methods. Kernel-based methods attempt to map the points into a space where they can be separated. The embedding function $\phi : \mathbb{R}^D \to \mathbb{R}^M$ maps points from the original space to an embedding space. Defining the embedding function ϕ is usually unknown and could be computationally expensive [1]. On the other hand, graph-based methods use the graph $G(V,E)$ whose set of vertices represents the data points and its set of edges represents the similarity between each pair of vertices. Finding non-convex clusters in a graph could be done in three ways: (1) by iteratively coarsening and partitioning the graph [2], (2) by performing spectral clustering [3], and (3) by feeding the graph $G(V,E)$ to a neural network (SpectralNet) [4]. The first way of detecting clusters in a graph is iterative which involves two deficiencies: the risk of being trapped in a local minima and the need for a stopping condition. This makes spectral clustering and SpectralNet more appealing for studies conducting graph-based clustering.

Spectral clustering starts by constructing a graph $G(V,E)$. The sets of vertices V and edges E represent data points and their pairwise similarities. Spectral clustering detects clusters by performing eigen-decomposition on the graph Laplacian matrix L and running k-means on its top eigenvectors [5]. The computational bottleneck represented by the eigen-decomposition would cost the algorithm computations in the order of $\mathcal{O}(N^3)$ [6]. This stimulated the research on reducing these computations by reducing the graph vertices and/or edges. However, the need for a memory efficient graph creates another problem related to the number of parameters associated with the process of graph construction. Deciding the number of reduced vertices and how the edges are eliminated would create several parameters that need careful tuning.

SpectralNet [4] uses Siamese nets to learn affinities between data points. Then it feeds these affinities to a neural network to find a map function F_θ, which maps the graph vertices V to an embedding space where they can be separated using k-means. The Siamese nets expect the user to label which pairs are positive (similar) and which are negative (dissimilar). An unsupervised pairing uses the k-nearest neighbors, where the nearest neighbors are the positive pairs and the farthest neighbors are the negative pairs. The parameter k requires manual tuning. It also restricts the number of edges to be exactly k, regardless of the surrounding density around the data point.

In her spectral clustering paper, von Luxburg [7] wrote about the advantages of mutual k-nearest neighbor graph, and how it *"tends not to connect areas of different density"*. She highlighted the need for having a *"heuristic to choose the parameter k"*. We introduce a graph reduction method that does not require any parameters to produce a mutual

* Corresponding author.
E-mail addresses: mashaan.alshammari@uoh.edu.sa (M. Alshammari), john.stavrakakis@sydney.edu.au (J. Stavrakakis), masa.takatsuka@sydney.edu.au (M. Takatsuka).

https://doi.org/10.1016/j.array.2022.100192
Received 21 January 2022; Received in revised form 14 May 2022; Accepted 22 May 2022
Available online 30 May 2022

graph with a reduced number of edges compared to the size of the full graph $E = N \times N$, where N is number of all vertices. It initially finds the mean distance that best describes the density around a point. Then, it computes the pairwise similarities based on: (1) the distance between a pair of points, and (2) the mean distance of the surrounding density. Finally, we construct a mutual graph where a pair of vertices must be in each other's nearest neighbors sets. We used two graph applications for the experiments: spectral clustering and SpectralNet. The proposed method provides a stable alternative compared to other methods where their performance was determined by the selected parameters.

Our main contribution in this work, is eliminating manually tuning parameters that affect the clustering accuracy when changed. The graph partitioning methods used in this work are spectral clustering [7] and SpectralNet [4].

2. Related work

The problem of detecting non-convex clusters has led to the development of numerous clustering methods. These methods have abandoned the assumption that a cluster has a single mean. Instead, they rely on pairwise similarities to detect clusters. Graph-based clustering involves two steps: (1) reducing the graph, and (2) partitioning the graph. The proposed method in this paper falls under graph construction methods.

Spectral clustering uses eigen-decomposition to map the points into an embedding space, then groups similar points. One of the important application of spectral clustering is subspace clustering [8,9]. The performance of spectral clustering is determined by the similarity metric used to construct the affinity matrix A. The earlier works of subspace clustering used affinities based on principal angles [10]. But recent studies have used sparse representation of points to measure the similarity [8,11,12]. Spectral clustering requires computations in order of $\mathcal{O}(N^3)$, due to the eigen-decomposition step. A straightforward solution to this problem is to reduce the size of the affinity matrix A. This can be done in two ways: (1) reducing the set of vertices V, or (2) reducing the set of edges E.

Reducing the number of vertices is done by placing representatives on top of the data points, and then using those representatives as graph vertices. Placing a representative could be done by sampling (like k-means++ [13]) or by vector quantization (like self-organizing maps [14]). A well-known method in this field is "k-means-based approximate spectral clustering (KASP)" proposed by Yan et al. [6]. KASP uses k-means to place representatives. Other efforts by Tasdemir [15] and Tasdemir et al. [16] involved placing representatives using vector quantization, and a nice feature of these methods is that the pairwise similarities are computed during the vector quantization. The problem with these methods is the parameter m, which is the number of representatives. Specifically, how should we set m? And how would different values of m affect the clustering results?

Reducing the graph edges could be done by setting the neighborhood conditions. For instance, let p be the center of a ball $B(p,r)$ with radius r and q be the center of a ball $B(q,r)$. p and q are connected if and only if the intersection of $B(p,r)$ and $B(q,r)$ does not contain other points [17]. Such graphs are called Relative Neighborhood Graphs (RNGs). Correa and Lindstrom [18] used a β-skeleton graph for spectral clustering. However, the parameter β needs tuning. Alshammari et al. [3] introduced a method to filter edges from a k-nearest neighbor graph. However, it still needs an influential parameter, which was the mean of the baseline distribution of distances μ_0. Another local method to reduce the number of edges was proposed by Satuluri et al. [19]. The authors measured the similarity between two vertices using adjacency lists overlap, a metric known in the literature as shared nearest neighbor similarity [20]. A graph sparsification method based on effective resistance was proposed by Spielman and Srivastava [21,22]. Their method was theoretically solid, but the definition of effective resistance breaks the cluster structure of the graph. Vertices with more short paths have low effective resistance and the method disconnects them. Retaining the cluster structure of the graph requires connecting such vertices [19].

In spectral clustering, the obtained spectral embedding cannot be extended to unseen data, a task commonly known as out-of-sample-extension (OOSE). Several studies have proposed solutions to this problem. Bengio et al. [23] used Nyström method to approximate the eigenfunction for the new samples. But they have to check the similarity between the training and new samples [24]. Alzate and Suykens [25] proposed binarizing the rows of eigenvectors matrix where each row corresponds to a single training data point. By counting row occurrences, one can find the k most occurring rows, where each row represents an encode vector for a cluster. To label a test sample, its projection was binarized and it is assigned to the closest cluster based on the minimum Hamming distance between its projection and encoding vectors. Levin et al. [26] proposed a linear least squares OOSE, which was very close to Bengio et al. [23] approach. They also proposed a maximum-likelihood OOSE that produces a binary vector $\overline{a}$ indicating whether the unseen sample has an edge to the training samples or not.

All previous methods that provided an out-of-sample-extension (OOSE) to spectral clustering have relied on eigen-decomposition, which becomes infeasible for large datasets. The newly proposed SpectralNet [4] is different from spectral clustering in a way that it does not use eigen-decomposition step. Instead, SpectralNet passes the affinity matrix A into a deep neural network to group points with high similarities. Yet, SpectralNet still needs a graph construction method. Previous SpectralNet works have used k-nearest neighbor graph, but they have to set the parameter k manually. It also restricts the number of edges to be exactly k, regardless of the surrounding density around the data point. Dense clusters require more edges to be strongly connected. Strong connections ensure closer positions in the embedding space. Also, SpectralNet methods randomly choose the negative pairs. This random selection makes the method inconsistent in independent executions.

Considering the literature on reduced graphs for spectral clustering and SpectralNet, it is evident that they have two deficiencies. First, certain parameters are required to drive the graph reduction process. Second, the involvement of random steps makes these methods inconsistent over independent executions.

3. Reducing the graph size without the need for parameters

The motivation behind our work was to avoid the use of any parameters during the graph reduction. The input for our method is a k-nearest neighbor graph. Although this k-nn graph is sparsified, it still connects clusters with different densities. The value of k has limited influence on the final graph because it is not final, and most of the unnecessary edges created by k-nn will be removed in the reduction process. The method starts by finding the value of σ_p that best describes the local statistics around a randomly chosen point p. Then, it filters the edges with low weights. Finally, it checks the mutual agreement for each edge.

3.1. Finding the value of σ_p

To compute pairwise similarities, we used the similarity measure introduced by [27], which is defined as follows:

$$A_{pq} = \exp\left(\frac{-d^2(p,q)}{\sigma_p \sigma_q}\right). \tag{1}$$

where $-d^2(p,q)$ is the distance between points p and q. σ_p and σ_q are the local scales at points p and q respectively. What is good about this similarity measure is that it uses two sources of information to compute the pairwise similarity: (1) the distance between them, and (2) the surrounding density for each point. Points belonging to clusters with different densities would have a low similarity even if they are

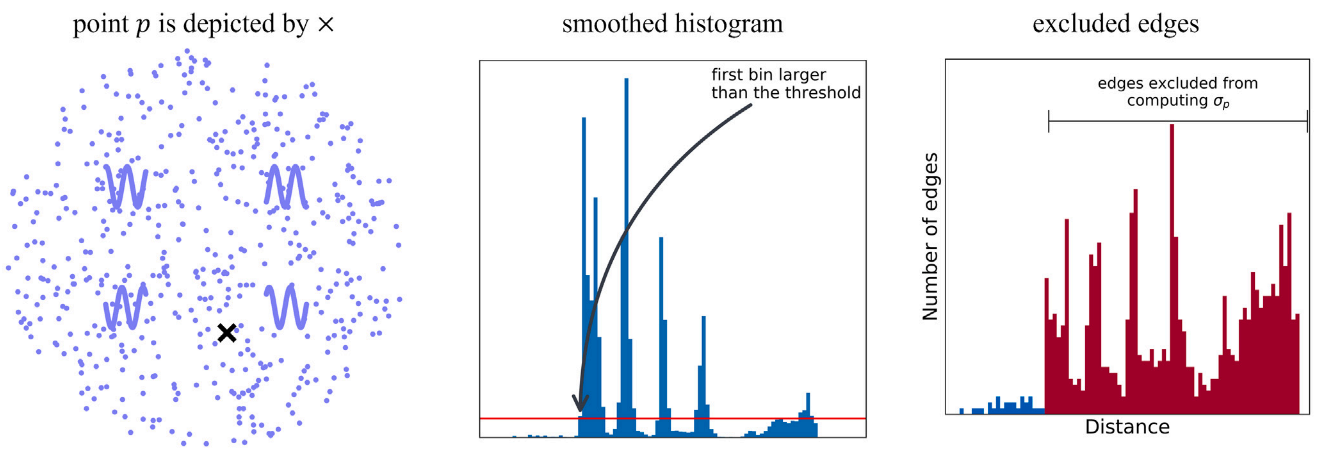

Fig. 1. The process of computing σ_p for a point p. (Best viewed in color).

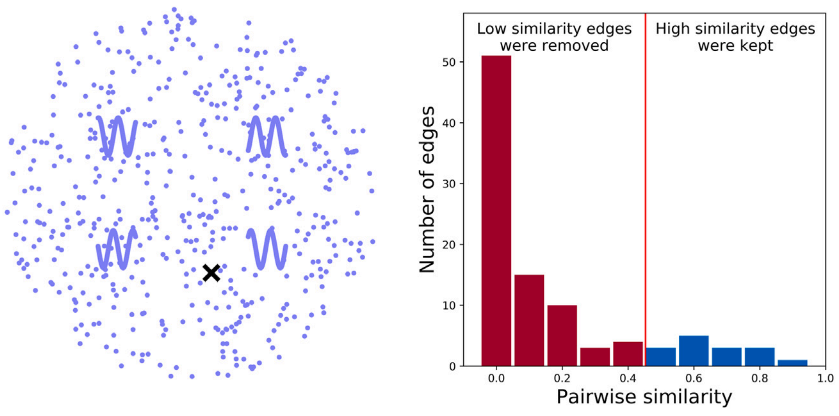

Fig. 2. After computing the pairwise similarities, we include highly similar edges for a point p. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

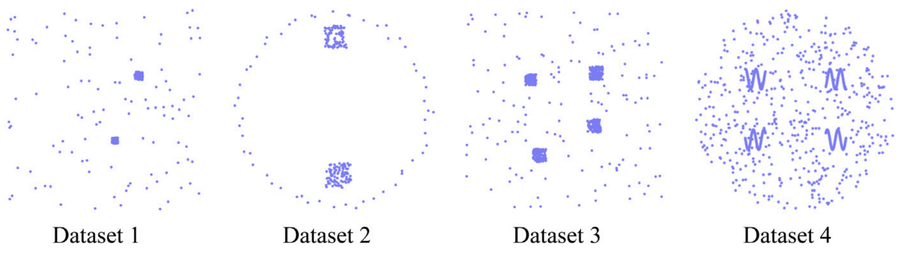

Fig. 3. Synthetic datasets used in the experiments.

separated by a small distance This makes this measure superior for highlighting different clusters separated by a small distance.

One problem that arises from using this measure in Eq. (1) is how to set the value of σ_p in the denominator. In previous studies, it was set as the distance to the 7th neighbor [27,28]. However, there is no evidence that the distance to the 7th neighbor would work in every dataset. Using the data to select this parameter would be more practical.

The idea behind the parameter σ_p is to measure the sparseness of a cluster. If p lies in a sparse cluster, it would have a large σ_p; whereas if p lies in a dense cluster, it would have a small σ_p. To achieve this, we need to exclude neighbors with different local density than p from being included in computing σ_p. We used a smooth histogram of distances to characterize the local density for p neighbors (as shown in Fig. 1). The intuition is if a neighbor has different local density than p, this would be represented as a peak on the histogram. The histogram bin values for each point are smoothed using the moving weighted average (MWA). The smoothing was designed as follows:

$$MWA_i = \frac{v_{i-1} + v_i + v_{i+1}}{r_{i-1} + r_i + r_{i+1}}, \tag{2}$$

where v is the value of the bin and r is the rank of the bin, with $r = 1$ being the bin containing the closest points to the point p. This smoothing assigns weights to the bins based on their distance from p, with high weights assigned to closer bins and low weights assigned to bins further away.

The histogram threshold tells us that up to the Kth neighbor, the local density of p has not changed. Then, we compute σ_p as the mean distance from the 1st to the Kth neighbor. This process is described in statements 4 to 9 in Algorithm 1.

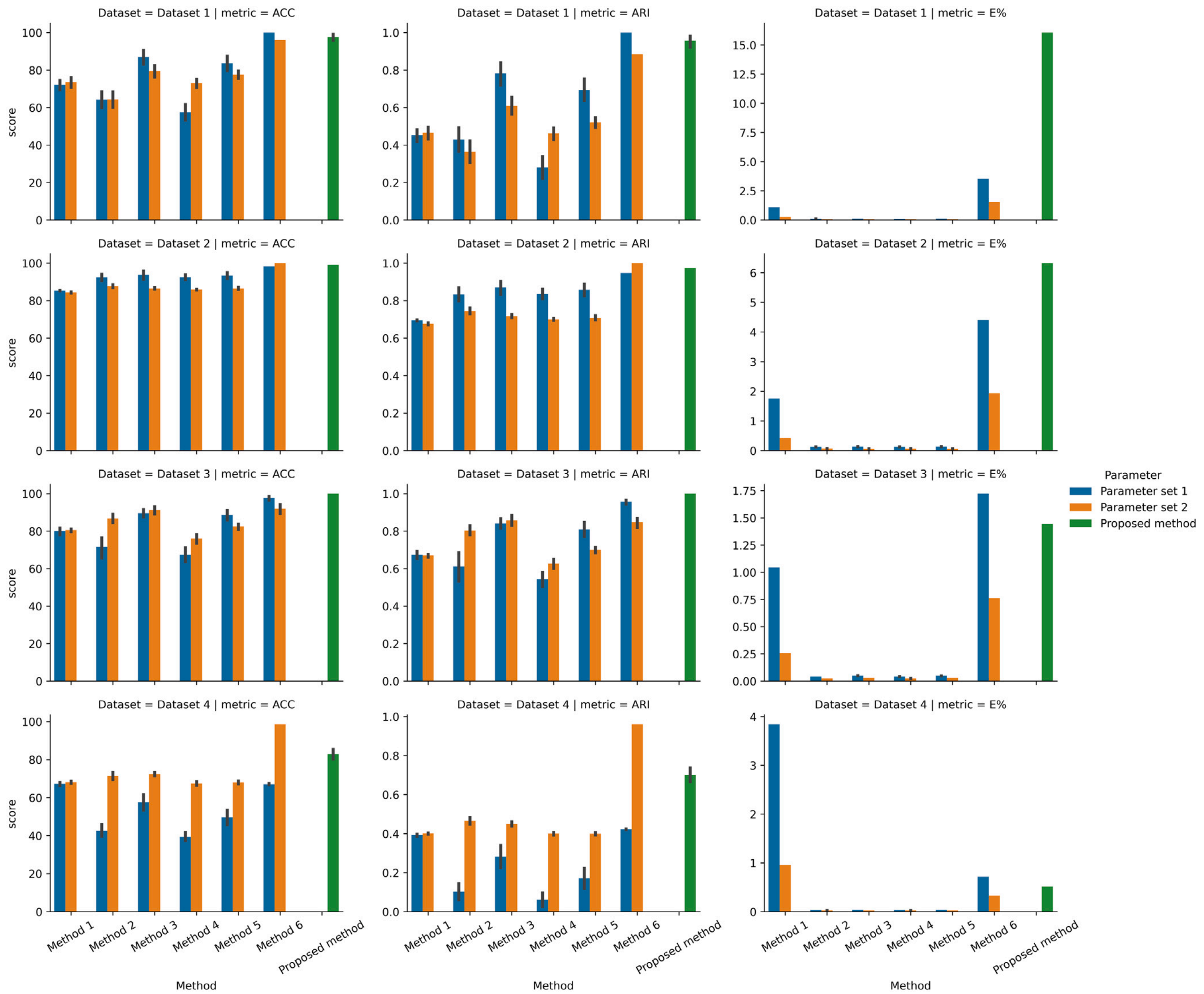

Fig. 4. Results with the synthetic data, all values are for 50 runs. (Best viewed in color).

3.2. Reducing the graph edges

Once we have σ_p for each point, we can calculate the pairwise similarities using the formula in Eq. (1), as shown in statements 10 to 14 in Algorithm 1. Large values indicate highly similar points, whereas low values indicate dissimilarity. We build another histogram of all the pairwise similarities using the Freedman–Diaconis rule [29] as shown in Fig. 2. For each point, similarities lower than the threshold T_p are eliminated. If the maximum similarity is larger than the mean plus the standard deviation $\mu + \sigma$, the threshold is set as $T = \mu + \sigma$. If not, the threshold is set as $T = \mu - \sigma$. Fig. 2 shows the included similarities as blue bins and the excluded similarities as red bins. The graph edges are defined as:

$$(p, q) \in E(G) \Leftrightarrow A_{pq} > T_p. \tag{3}$$

where (p, q) is the edge between points p and q. A_{pq} is the weight assigned to the edge (p, q). This process is described in statements 15 to 21 in Algorithm 1. The last step of our reduction method is to build a mutual graph. In a mutual graph, a pair of points should agree to accept an edge. This makes the graph G to be defined as:

$$(p, q) \in E(G) \Leftrightarrow A_{pq} > T_p \quad \text{and} \quad A_{qp} > T_q. \tag{4}$$

where T_p is threshold of acceptance for the vertex p.

3.3. Integration with SpectralNet

Our graph filtering method can be seamlessly integrated to the newly proposed spectral clustering using deep neural networks (SpectralNet) [4]. SpectralNet uses Siamese nets [30] to learn affinities between data points. Siamese nets expect the user to label which pairs are positive and which are negative. An unsupervised pairing uses the k-nearest neighbors, where the nearest neighbors are positive pairs and the farthest neighbors are negative pairs. Our graph filtering can be used to obtain positive and negative pairs. It offers the advantage of setting the number of pairs per point dynamically. This cannot be achieved using k-nearest neighbors, where all the points are restricted to have a fixed number of positive pairs. Also, we do not have to set k

Algorithm 1: Reducing a k-nearest neighbor graph

```
   Input: k-nn graph where $k = k_{max}$ of $N$ vertices.
   Output: Reduced graph of $N$ vertices.
1  Construct distance matrix $D(N, k_{max})$ of $k$-nn graph
2  Construct a histogram $H_D$ of all elements in $D$ using FD rule
3  Save bin width in $H_D$ to the variable $bin_D$
   /* The following loop has computations in order of
      $\mathcal{O}(Nk_{max})$                                  */
4  for $p = 1$ to $N$ do
5      Construct a histogram $H_p$ of $D_{p,1 \text{ to } k_{max}}$ using $bin_D$
6      Apply MWA to bin values in $H_p$ (Eq. (2))
7      Set Kth as the first bin that exceeds MWA threshold
8      $\sigma_p = \text{mean}(D_{p,1 \text{ to } K^{th}})$
9  end
   /* The following loop has computations in order of
      $\mathcal{O}(Nk_{max})$                                  */
10 for $p = 1$ to $N$ do
11     for $q = 1$ to $k_{max}$ do
12         $A_{p,q} = \exp\left(\frac{D(p,q)}{\sigma_p \sigma_q}\right)$
13     end
14 end
   /* The following loop has computations in order of
      $\mathcal{O}(Nk_{max})$                                  */
15 for $p = 1$ to $N$ do
16     if $max(A_{p,1 \text{ to } k_{max}}) > \mu(A_{p,1 \text{ to } k_{max}}) + \sigma(A_{p,1 \text{ to } k_{max}})$ then
17         $A_{p,1 \text{ to } k_{max}} < \mu(A_{p,1 \text{ to } k_{max}}) + \sigma(A_{p,1 \text{ to } k_{max}}) = 0$
18     else
19         $A_{p,1 \text{ to } k_{max}} < \mu(A_{p,1 \text{ to } k_{max}}) - \sigma(A_{p,1 \text{ to } k_{max}}) = 0$
20     end
21 end
22 Construct a reduced graph using affinity matrix $A(N, k_{max})$
```

manually. We let our method assigns the positive and negative pairs for each point. A pseudocode illustrating the steps of the proposed method is shown in Algorithm 1.

4. Experiments and discussions

In the experiments we used four synthetic datasets, as shown in Fig. 3. Dataset 1 to 3 were created by [27], while Dataset 4 was created by us. We also used seven real datasets (see Table 1). Apart from the MNIST dataset, all the real datasets were retrieved from UCI machine learning. Each dataset was run with two parameter sets to evaluate the effect.

Six methods were used for comparison and these are shown in Table 2. Methods 1 to 5 [6,15,16] rely on the parameter m, which is the number of representatives to build the graph G. They used iterative algorithms like k-means and self-organizing maps to construct the graph, which makes them produce a slightly different graph with each run. Method 6 [3] relied on the parameter μ_0 to build the graph G, where μ_0 is the number of neighbors whose mean was used as a threshold to include or exclude further neighbors. The code is available at https://github.com/mashaan14/Spectral-Clustering.

All the methods were evaluated using three evaluation metrics: (1) clustering accuracy (**ACC**) (2) the Adjusted Rand Index (**ARI**) [31], and (3) the percentage of edges used compared to all the edges in a full graph (**E%**).

ACC computes the percentage of hits between ground truth labels T_i and labels obtained through clustering L_i. It is defined as [32]:

$$ACC(T, L) = \frac{\sum_{i=1}^{N} \delta(T_i, map(L_i))}{N}, \tag{5}$$

Table 1
The four synthetic and seven real datasets used in the experiments; N is the number of points, d is the number of dimensions, C is the number of clusters, m is the size of the reduced set of vertices, and μ_0 is the number of neighbors used as a threshold to include or exclude further neighbors.

Synthetic datasets							
				Parameter set 1		Parameter set 2	
	N	d	C	m	μ_0	m	μ_0
Dataset 1	303	2	3	32	7	16	3
Dataset 2	238	2	3	32	7	16	3
Dataset 3	622	2	5	64	7	32	3
Dataset 4	1304	2	5	256	7	128	3
Real datasets							
				Parameter set 1		Parameter set 2	
	N	d	C	m	μ_0	m	μ_0
iris	150	4	3	32	7	16	3
wine	178	13	3	32	7	16	3
BC-Wisc.	699	9	2	40	7	20	3
statlog	6435	5	6	100	7	50	3
MNIST	10000	100	10	500	7	100	3
PenDigits	10992	16	10	500	7	100	3
mGamma	19020	10	2	2000	7	1000	3

where N is the number of points and the function $\delta(x, y)$ is the Kronecker delta function, which equals one if $x = y$ and zero otherwise. The function $map(L_i)$ permutes the grouping obtained through clustering for the best fit with the ground-truth grouping. ARI needs two groupings T and L, where T is the ground truth and L is the grouping predicted by the clustering method. If T and L are identical, ARI produces one, and zero in case of random grouping. ARI is calculated using: n_{11}: pairs in the same cluster in both T and L; n_{00}: pairs in different clusters in both T and L; n_{01}: pairs in the same cluster in T but in different clusters in L; n_{10}: pairs in different clusters in T but in the same cluster in L.

$$ARI(T, L) = \frac{2(n_{00}n_{11} - n_{01}n_{10})}{(n_{00} + n_{01})(n_{01} + n_{11}) + (n_{00} + n_{10})(n_{10} + n_{11})}. \tag{6}$$

The computational efficiency can be measured by the method's running time, but, this is influenced by the type of machine used. We chose to measure the computational efficiency by the percentage of edges, **E%**:

$$E\% = \frac{E(G_{reduced})}{E(G_{full})}. \tag{7}$$

4.1. Experiments on synthetic data

In the synthetic datasets the proposed method delivered a performance that ranked it as 2nd, 2nd, 1st, and 2nd for Datasets 1 to 4 respectively (see Fig. 4). Method 6 was the top performer on three occasions. However, its performance dropped significantly when we changed the parameter μ_0. For example, its performance dropped by 50% with Dataset 4 when we changed $\mu_0 = 3$ to $\mu_0 = 7$. This shows how parameters could affect the performance. Another observation is the consistency of ACC and ARI metrics over the 50 runs. By looking at Fig. 4, the methods 1 to 5 have a wide standard deviation. This is explained by the iterative algorithms used by methods 1 to 5 to construct the graph. Method 6 and the proposed method do not have this problem, and they have a small standard deviation. This is due to their deterministic nature when constructing the graph, which makes them consistent over independent executions.

In terms of the used edges, the proposed method used 6.32%, 1.45%, and 0.51% of the full graph edges for Datasets 2 to 4 respectively. But in Dataset 1 there was a sharp increase where the proposed method used 16% of the full graph edges. This sharp increase could be explained by the points in dense clusters being fully connected.

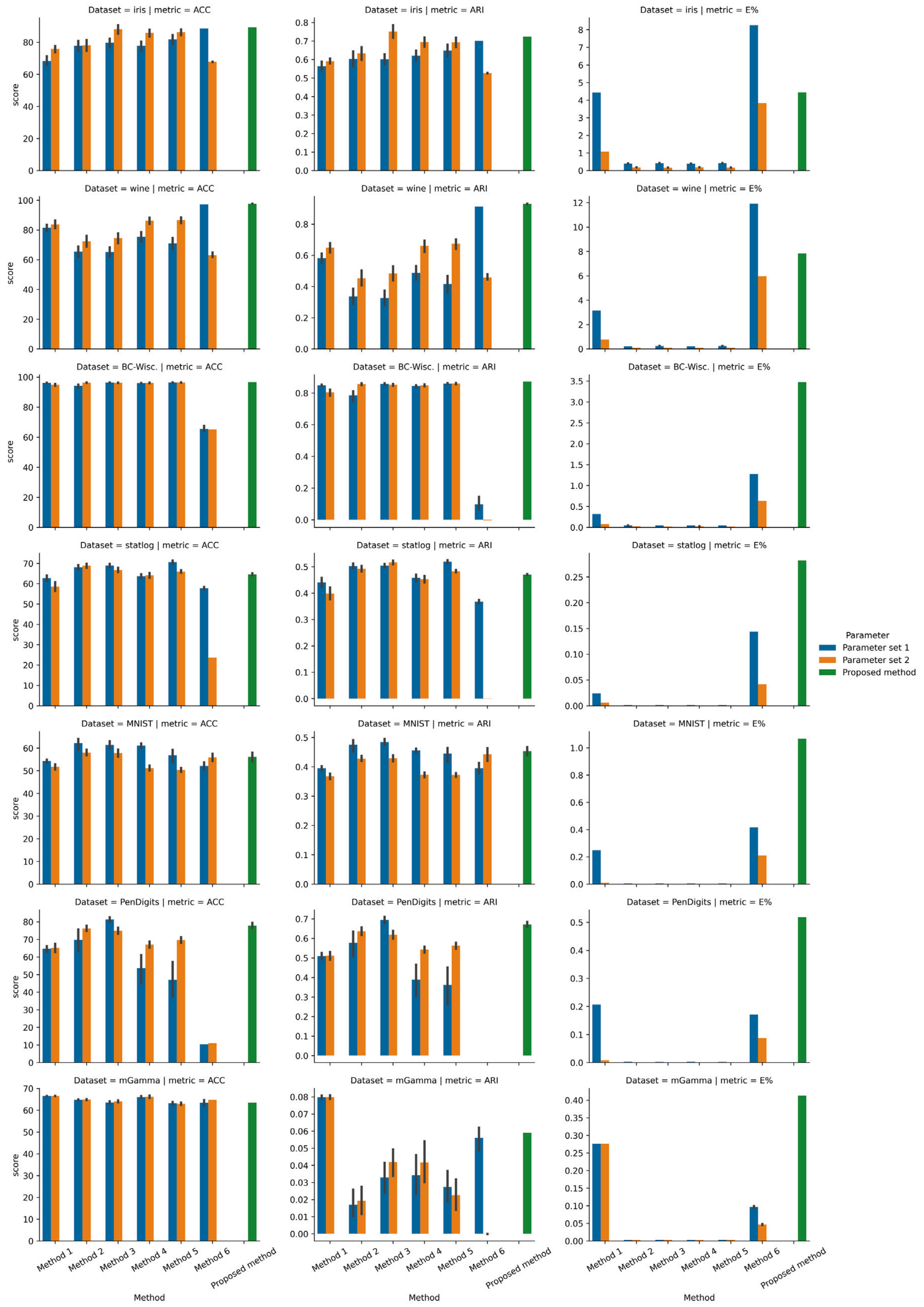

Fig. 5. Results with real data, all values are for 50 runs. (Best viewed in color).

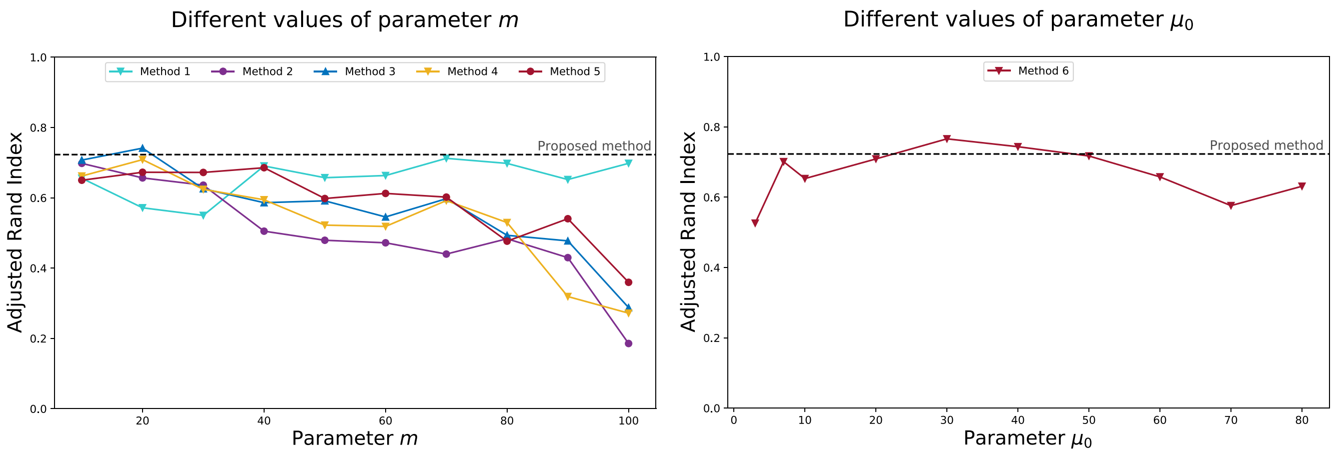

Fig. 6. Testing the methods' performance with the `iris` dataset under different settings of parameters m and μ_0. (Best viewed in color).

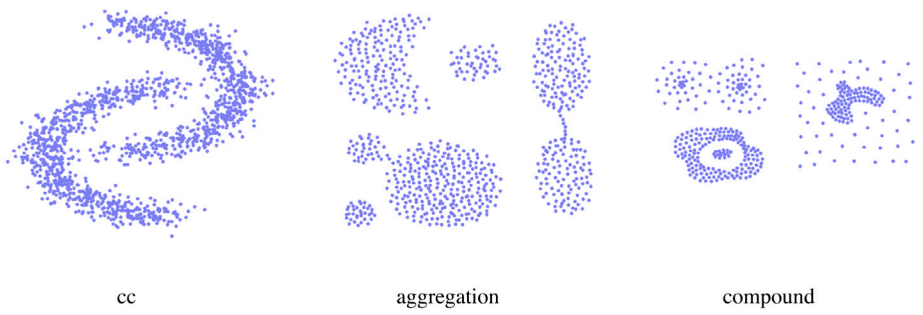

Fig. 7. Datasets used in the SpectralNet experiments.

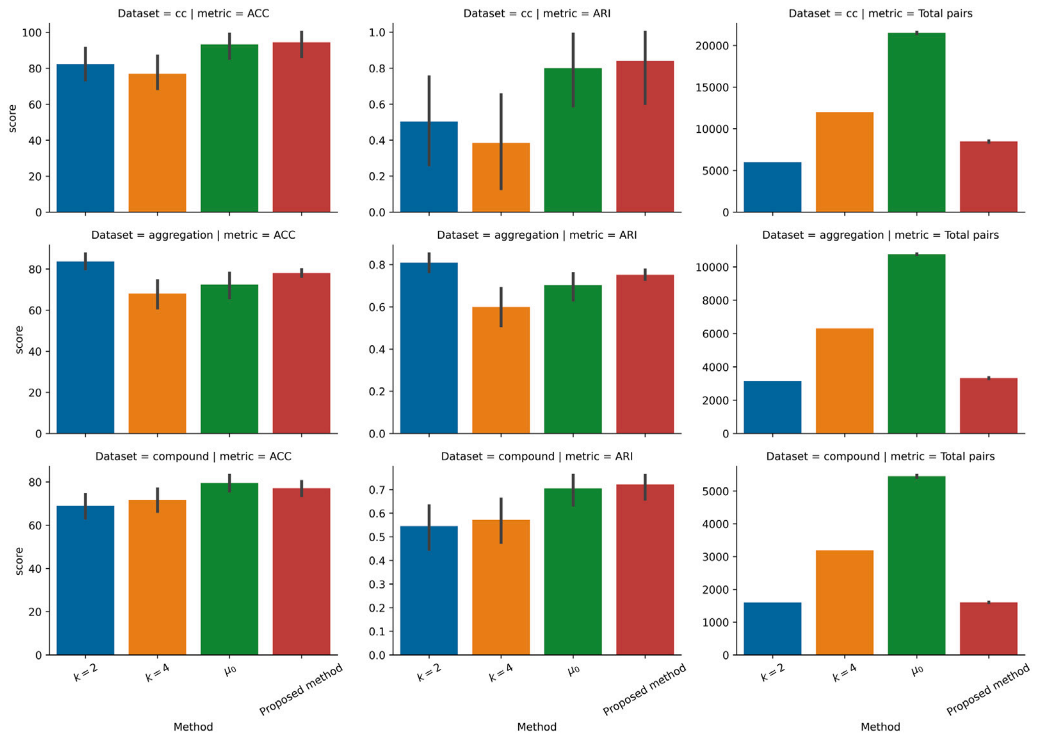

Fig. 8. Results of the experiments for integration with SpectralNet for 10 runs. (Best viewed in color).

Table 2
Methods used in the experiments. m is the number of reduced vertices, N is the number of all vertices, t is the number of iterations, k_{max} is the parameter used to construct k-nn graph.

	Method 1	**Method 2**	**Method 3**	**Method 4**	**Method 5**	**Method 6**	**Proposed method**
Reference	Yan, et al. (2009)	Tasdemir (2012)	Tasdemir (2012)	Tasdemir, et al. (2015)	Tasdemir, et al. (2015)	Alshammari, et al. (2021)	
Vertices reduction method	k-means	k-means	SOM	k-means	SOM	N/A	N/A
Similarity measure	Local σ	CONN	CONN	CONN Hybrid	CONN Hybrid	Local σ	Local σ
Graph construction time complexity	$O(mNt)$	$O(mNt)$	$O(m^2)$	$O(mNt)$	$O(m^2)$	$O(Nk_{max})$	$O(Nk_{max})$
Iterative graph construction?	Yes	Yes	Yes	Yes	Yes	No	No
Spectral clustering time complexity	$O(m^3)$	$O(m^3)$	$O(m^3)$	$O(m^3)$	$O(m^3)$	$O(N^3)$	$O(N^3)$
Influential parameter	m	m	m	m	m	μ_0	N/A

4.2. Experiments on real data

With real datasets in Fig. 5, the proposed method continued to be the most consistent method over all tested methods. It kept a very small standard deviation, while other methods had a wide standard deviation. The performance of other methods was determined by their parameters. For example, `Method 3` was the best performer on `iris` dataset when $m = 16$. However, when we changed m to 32, its performance dropped by more than 15%. Another observation with `statlog` and `MNIST` the proposed method did not perform well. This indicated that a cluster in these datasets does not have the same statistics across its regions. Therefore, characterizing clusters using local σ might not be a good choice. Instead, we should use CONN to discover clusters discontinuity, rather than tracking local statistics.

4.3. Effect of the parameters on the spectral clustering performance

In this experiment, we investigated how a wide selection of parameters could affect the accuracy of the spectral clustering. The parameters m and μ_0 were given the following values: $m \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and $\mu_0 \in \{3, 7, 10, 20, 30, 40, 50, 60, 70, 80\}$. In Fig. 6 (left), the performance `Methods 1` to `5` fluctuated with different values of m, with a clear downward trend seen as m increased. The dashed horizontal line is the performance of the proposed method. In Fig. 6 (right), `Method 6` started with low performance, peaking around $\mu_0 = 30$, and then it took a downward trend. By eliminating the use of μ_0, our method delivered a stable performance as shown by the horizontal dashed line.

4.4. Experiments for integration with SpectralNet

The SpectralNet integration experiment was conducted using three datasets shown in Fig. 7. The evaluation metrics are **ACC** shown in Eq. (5), **ARI** shown in Eq. (6), and **total pairs**, which is the number of pairs passed to the Siamese net. We used four methods to construct positive and negative pairs. The first two methods used a k-nearest neighbor graph with $k = 2$ and $k = 4$. Simply, the nearest k neighbors were set as the positive pairs, and k random farthest neighbors were set as the negative pairs. The third method used the parameter μ_0 proposed by Alshammari, et al. [3] to construct pairs.

In Fig. 8, the proposed method delivered the best performance for the `cc` and `compound` datasets. This good performance was coupled with good computational efficiency, with an average of 8468 for the total pairs passed to the Siamese net. Only $k = 2$ could deliver fewer total pairs, but with a massive loss in performance. For the `aggregation` dataset, $k = 2$ delivered the best performance. This experiment highlighted the need for setting the number of positive pairs dynamically. The methods following this approach (the μ_0 method and our method) were the best performers for two of the three datasets.

5. Conclusion

The problem of detecting non-convex clusters has led to the development of numerous clustering methods. One of the well-known graph-based clustering methods is spectral clustering and SpectralNet. Both spectral clustering and SpectralNet require a graph that connects points in the same cluster with edges of high weights. The intuition is simple, strongly connected points will become closer in the embedding space and can be easily detected.

Graph reduction requires extensive use of parameters that need careful setting for each dataset. The graph reduction algorithm proposed in this study does not require any parameters to reduce the graph, yet it is able to maintain spectral clustering and SpectralNet accuracies. It takes an input as a full graph or a k-nearest neighbor graph (in the case of a large number of points). Then, it reduces the graph edges using statistical measures that require low computations. The experiments revealed that the proposed method provides a stable alternative compared to other methods that require parameters tuning.

The proposed method does not reduce the graph vertices, which could boost the computational efficiency. A useful extension of the proposed method would be a vertices reduction component that is aware of local statistics. Another potential improvement of this work is to use a different kernel other than gaussian kernel to compute pairwise similarities.

CRediT authorship contribution statement

Mashaan Alshammari: Conceptualization, Methodology, Software, Visualization, Writing – original draft, Project administration. **John Stavrakakis:** Conceptualization, Investigation, Visualization, Writing – review & editing. **Masahiro Takatsuka:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Qin Y, Yu ZL, Wang C-D, Gu Z, Li Y. A novel clustering method based on hybrid K-nearest-neighbor graph. Pattern Recognit 2018;74:1–14. http://dx.doi.org/10.1016/j.patcog.2017.09.008.

[2] Kim Y, Do H, Kim SB. Outer-points shaver: Robust graph-based clustering via node cutting. Pattern Recognit 2020;97:107001. http://dx.doi.org/10.1016/j.patcog.2019.107001.

[3] Alshammari M, Stavrakakis J, Takatsuka M. Refining a k-nearest neighbor graph for a computationally efficient spectral clustering. Pattern Recognit 2021;114:107869. http://dx.doi.org/10.1016/j.patcog.2021.107869.

[4] Shaham U, Stanton K, Li H, Nadler B, Basri R, Kluger Y. SpectralNet: Spectral clustering using deep neural networks. In: 6th international conference on learning representations, ICLR 2018 - Conference track proceedings. 2018, http://dx.doi.org/10.48550/ARXIV.1801.01587.

[5] Ng AY, Jordan MI, Weiss Y. On spectral clustering: Analysis and an algorithm. Adv Neural Inf Process Syst 2002.

[6] Yan D, Huang L, Jordan MI. Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. 2009, p. 907–16. http://dx.doi.org/10.1145/1557019.1557118.

[7] von Luxburg U. A tutorial on spectral clustering. Stat Comput 2007;17(4):395–416. http://dx.doi.org/10.1007/s11222-007-9033-z.

[8] Liu G, Lin Z, Yan S, Sun J, Yu Y, Ma Y. Robust recovery of subspace structures by low-rank representation. IEEE Trans Pattern Anal Mach Intell 2013;35(1):171–84. http://dx.doi.org/10.1109/TPAMI.2012.88.

[9] Elhamifar E, Vidal R. Sparse subspace clustering: Algorithm, theory, and applications. IEEE Trans Pattern Anal Mach Intell 2013;35(11):2765–81. http://dx.doi.org/10.1109/TPAMI.2013.57.

[10] Wolf L, Shashua A. Learning over sets using kernel principal angles. J Mach Learn Res 2003;4(null):913–31. http://dx.doi.org/10.1109/TPAMI.2012.88.

[11] Peng C, Kang Z, Li H, Cheng Q. Subspace clustering using log-determinant rank approximation. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '15, 2015, p. 925–34. http://dx.doi.org/10.1145/2783258.2783303.

[12] Peng C, Zhang Q, Kang Z, Chen C, Cheng Q. Kernel two-dimensional ridge regression for subspace clustering. Pattern Recognit 2021;113:107749. http://dx.doi.org/10.1016/j.patcog.2020.107749.

[13] Arthur D, Vassilvitskii S. K-means++: The advantages of careful seeding. In: Proceedings of the annual ACM-SIAM symposium on discrete algorithms 07-09-January-2007. 2007, p. 1027–35.

[14] Kohonen T. The self-organizing map. Proc IEEE 1990;78(9):1464–80. http://dx.doi.org/10.1109/5.58325.

[15] Tasdemir K. Vector quantization based approximate spectral clustering of large datasets. Pattern Recognit 2012;45(8):3034–44. http://dx.doi.org/10.1016/j.patcog.2012.02.012.

[16] Tasdemir K, Yalcin B, Yildirim I. Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures. Pattern Recognit 2015;48(4):1465–77. http://dx.doi.org/10.1016/j.patcog.2014.10.023.

[17] Marchette DJ. Random graphs for statistical pattern recognition. Wiley series in probability and statistics, Hoboken, N.J: Wiley-Interscience; 2004, http://dx.doi.org/10.1002/047172209X.

[18] Correa C, Lindstrom P. Locally-scaled spectral clustering using empty region graphs. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. 2012, p. 1330–8. http://dx.doi.org/10.1145/2339530.2339736.

[19] Satuluri V, Parthasarathy S, Ruan Y. Local graph sparsification for scalable clustering. SIGMOD '11, New York, NY, USA: Association for Computing Machinery; 2011, p. 721–32. http://dx.doi.org/10.1145/1989323.1989399.

[20] Jarvis R, Patrick E. Clustering using a similarity measure based on shared near neighbors. IEEE Trans Comput 1973;C-22(11):1025–34. http://dx.doi.org/10.1109/T-C.1973.223640.

[21] Spielman DA, Srivastava N. Graph sparsification by effective resistances. SIAM J Comput 2011;40(6):1913–26. http://dx.doi.org/10.1137/080734029.

[22] Spielman DA, Teng S-H. Spectral sparsification of graphs. SIAM J Comput 2011;40(4):981–1025. http://dx.doi.org/10.1137/08074489X.

[23] Bengio Y, Paiement J-f, Vincent P, Delalleau O, Roux N, Ouimet M. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In: Thrun S, Saul L, Schölkopf B, editors. Advances in neural information processing systems, Vol. 16. MIT Press; 2003.

[24] Nie F, Zeng Z, Tsang IW, Xu D, Zhang C. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. IEEE Trans Neural Netw 2011;22(11):1796–808. http://dx.doi.org/10.1109/TNN.2011.2162000.

[25] Alzate C, Suykens JAK. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. IEEE Trans Pattern Anal Mach Intell 2010;32(2):335–47. http://dx.doi.org/10.1109/TPAMI.2008.292.

[26] Levin K, Roosta F, Mahoney M, Priebe C. Out-of-sample extension of graph adjacency spectral embedding. In: Dy J, Krause A, editors. Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research, vol. 80, PMLR; 2018, p. 2975–84.

[27] Zelnik-Manor L, Perona P. Self-tuning spectral clustering. Adv Neural Inf Process Syst 2005;1601–8.

[28] Sugiyama M. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J Mach Learn Res 2007;8(May):1027–61.

[29] Freedman D, Diaconis P. On the histogram as a density estimator L2 theory. Z Wahrscheinlichkeitstheor Verwandte Geb 1981;57(4):453–76. http://dx.doi.org/10.1007/BF01025868.

[30] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a "Siamese" time delay neural network. In: Proceedings of the 6th international conference on neural information processing systems. NIPS'93, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1993, p. 737–44. http://dx.doi.org/10.1142/9789812797926_0003.

[31] Hubert L, Arabie P. Comparing partitions. J Classification 1985;2(1):193–218. http://dx.doi.org/10.1007/BF01908075.

[32] Cai D, He X, Han J. Document clustering using locality preserving indexing. IEEE Trans Knowl Data Eng 2005;17(12):1624–37. http://dx.doi.org/10.1109/TKDE.2005.198.

Dr. Mashaan Alshammari is an assistant professor at University of Hail. His research interests include unsupervised learning and image analysis. Mashaan holds a MSc in computer science from King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, and a PhD from the University of Sydney, Australia.

Dr. John Stavrakakis has strong interests in 3D computer graphics, remote rendering and computer security. He holds a PhD in Computer science and is an academic fellow at the University of Sydney, Australia.

Dr. Masahiro Takatsuka received his MEng degree at Tokyo Institute of Technology in 1992, and received his PhD at the Monash University in 1997. In 1997–2002, he worked at GeoVISTA Center, The Pennsylvania State University as a senior research associate. He joined the School of Computer Science, the University of Sydney in 2002.