A parameter-free graph reduction for efficient spectral clustering

Mashaan Alshammari*, John Stavrakakis, Masahiro Takatsuka

School of Computer Science, The University of Sydney, NSW 2006, Australia

Abstract

Conventional clustering methods like k-means fail to detect clusters of non-convex shape. This led to the development of several clustering methods that can detect non-convex clusters. Spectral clustering is one of these methods. It models data points as vertices V on a graph G and maps them into a space where they can be separated. The mapping is performed through eigen-decomposition which requires heavy computations of $O(n^3)$. Many studies have proposed algorithms to reduce the size of the graph G to reduce computations. But the focus on reducing the graph size has led to an extensive use of parameters to be tuned for each dataset. We introduce a graph reduction algorithm that does not require any parameters. It takes a full graph as an input. It uses statistical measures to reduce graph edges to then be used for eigen-decomposition. Experiments showed that our method provides a stable alternative, where other methods' performance fluctuates according to the setting of their parameters.

Keywords: Spectral clustering, Graph-based clustering, Graph reduction, Local scale similarity

*Corresponding author. Tel.: +61 2 9351 3423; fax: +61 2 9351 3838
Email address: mals6571@uni.sydney.edu.au (Mashaan Alshammari)

1. Introduction

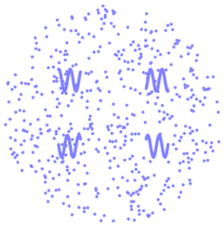
Unsupervised classification, which is also known as clustering, refers to the problem of partitioning N points $\{1, 2, ..., N\}$ into C disjoint subsets $\{1, 2, ..., C\}$ without a prior knowledge of their categorization [1]. In supervised and semi supervised classification, the method predicts the category of a sample based on known categories around it and statistical measures such as bayesian rule. This is not the case in unsupervised classification where the method predicts the category of a sample only based on statistical measures. Most well-known clustering methods including k-means [2] and generative mixture models [3] would only detect clusters of convex geometric shapes [4]. Thus, they fail when this assumption is violated. That is because their statistical measures are looking for the cluster mean with samples scattered around it.

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-based and graph-based methods. Kernel-based methods attempt to map the points into a space where they can be separated. The embedding function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$ maps points from the original space to an embedding space [5]. Defining the embedding function ϕ is usually unknown and could be computationally expensive [6]. On the other hand, graph-based methods use the graph $G(V, E)$ whose set of vertices represent data points and its set of edges represent the similarity between each pair of vertices. Finding non-convex clusters in a graph could be done in two ways: 1) by iteratively coarsening and partitioning the graph [6, 7, 8], and 2) by performing spectral clustering [9, 10, 11].

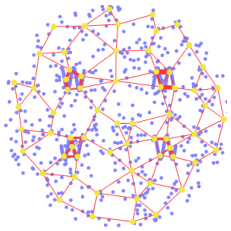
The former way is iterative which involves two deficiencies: the risk of being trapped in a local minima and the need for a stopping condition. This makes spectral clustering more appealing for studies conducting graph-based clustering.

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 [12]. The computational bottleneck represented by the eigen-decomposition would cost the algorithm computations in the order of $O(n^3)$ [13]. 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.

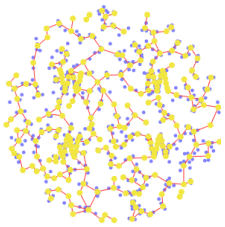
We introduce a graph reduction that does not require any parameters to reduce the graph size (see Fig.1). The reduced graph is very light compared to the size of full graph $E = N \times N$. The proposed reduction is a data-driven process. It initially finds the mean distance that best describe the density around a point. This parameter is usually set in the literature as the distance to the 7^{th} neighbor [14, 15]. Then, it computes the pairwise similarity (i.e., a graph edge) according to: 1) distance between the pair of points, and 2) the mean distance of surrounding density. Each point is tested individually to exclude any edge value that seems to be an outlier for this point. If a



Original data

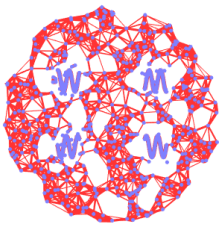


$m = 64$

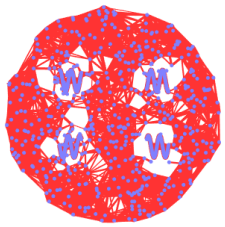


$m = 256$

An attempt to reduce the graph vertices with two selections of the parameter m. m is the size of the reduced set of vertices.

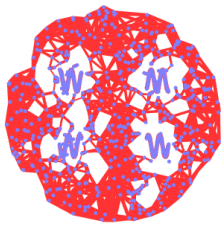


$\mu_0 = 7$



$\mu_0 = 21$

An attempt to reduce the graph edges with two selections of the parameter μ_0. μ_0 is the number of neighbors used as a threshold to include or exclude further neighbors.



Reducing the graph size without any parameters

Fig. 1. Different methods attempt to reduce the graph size (best viewed in color).

pair of points decide to keep the edge connecting them, that edge would have mutual agreement. Edges with no mutual agreement are excluded from the final graph. The result is a graph which only connects points with close distance and share similar surrounding density.

The outline of this paper as follows: section 2 provides an overview of methods used to detect non-convex clusters. Section 3 introduces the proposed method and its steps to reduce edges in the graph. In section 4 we describe the setup of the experiments and discuss the findings.

2. Related work

The problem of detecting non-convex clusters has led to the development of numerous clustering methods. These methods abandoned the assumption that a cluster has a single mean. Instead, these methods relied on pairwise similarities to detect clusters. The following subsections provide an overview about graph-based clustering.

2.1. DBSCAN-like methods

The Density-based spatial clustering of applications with noise (DBSCAN) [16] is a well-known method to detect non-convex clusters. It is a graph-based clustering method. DBSCAN uses two parameters, $MinPts$ which is the minimum number of neighbors in a region of size Eps. Based on these parameters, DBSCAN classify points into core, shared, and noise points. However, selecting parameters is not trivial and global setting of parameters might not be suitable for some parts of data [17].

There were efforts to derive DBSCAN parameters from the data. Ertöz et al. [18] merged Jarvis-Patrick algorithm (also known as shared nearest

neighbor) [19] with DBSCAN. It redefines the parameter *Eps* to be SNN density and identify core points accordingly. A similar approach was adopted by Inkaya et al. [20] and Inkaya [21]. It assigns direct neighbors as core neighbors. Then, it measures the similarity between a pair of points based on the number of shared core points.

A recent work by Kim et al. [7] proposed a "pseudo-density vector" that estimates the importance of a point to reconstruct the entire dataset. Points between clusters or on cluster boundary would have a low pseudo-density vector and called "outer points". This allows the algorithm to: 1) filter outer points, 2) cluster based on core points, and 3) assign outer points to their nearest clusters. Another effort by Li et al. [8] attempts to find clusters by linking subgraphs in dense regions and filtering out subgraphs in sparse regions. The problem with these approaches is the assumption that clusters are dense which means sparse clusters will be filtered out and labeled as noise.

The iterative nature of DBSCAN-like methods requires them to have a starting point and a stopping criterion. Such methods usually start by building subgraphs, then examine if linking these graphs is possible. It is unclear if different starting points would lead to the same clustering result. Also, traversing the graph in an attempt to link subgraphs could be costly in terms of computations.

2.2. Spectral clustering

Spectral clustering uses a graph cut approach to find non-convex clusters. It could be traced back to early works in 1990s [22, 23]. However, its most used algorithm is the one provided by Ng et al. [12]. It starts by building a

graph $G(V, E)$ represented by the affinity matrix A:

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

Then, the affinity matrix A and the degree matrix $D_{pp} = \sum_q A_{qp}$ are used to compute the graph Laplacian L [24]:

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}. \tag{2}$$

The top eigenvectors of L represent a mapped space where clusters can be separated by k-means.

The eigen-decomposition step is the computational bottleneck for spectral clustering with computations of $O(n^3)$. The size of A highly influences the spectral computations. Therefore, most studies have focused on making A smaller or sparser by removing vertices or edges from the graph G. Less vertices mean A would be smaller with size $m \times m$ where $m \ll N$. Less edges mean more zero entries, allowing eigen-decomposition to run on a sparse version of L. Making A smaller or sparser would serve the goal of speeding up spectral computations.

Reducing number of vertices is done by placing representatives on top of data points, then use these representatives as graph vertices. Placing representative could be done by sampling (like k-means++ [25]) or vector quantization (like k-means [25] and self-organizing maps [26]). A well-known method in this field is "k-means-based approximate spectral clustering" KASP by Yan et al. [13]. KASP uses k-means to place representatives. Other efforts by Tasdemir [27] and Tasdemir et al. [28] to place representa-

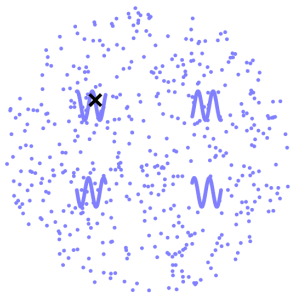
tives using vector quantization. A nice feature about their methods is that pairwise similarities are computed during vector quantization. We also have a couple of studies in this field [29, 30]. The problem with these methods is the parameter m that is the number of representatives. How to set m? and how different values of m would affect the clustering result.

Reducing graph edges could be done by setting neighborhood conditions. If the region between two points satisfy a predefined condition an edge would be drawn between them. Such graphs are called proximity graphs. Correa and Lindstrom [31] used a β-skeleton graph for spectral clustering.. However, the parameter β needs tuning. Alshammari et al. [32] introduced a method to filter edges from k-nearest neighbor graph. Nevertheless, it still needs an influential parameter that is the mean of baseline distribution of distances μ_0. Considering the literature on reduced graphs, it is evident that parameters are required to drive the graph reduction process

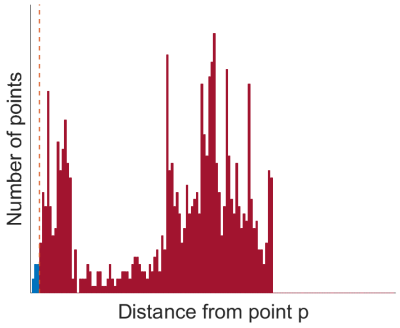
3. Reducing the graph size without the need of parameters

The motivation behind our work is to avoid the use of any parameters during graph reduction. The input for our method is a k-nearest neighbor graph. The value of k has a limited influence on the final graph, in fact we used $k = N - 1$ for most datasets. The method starts by finding the value of σ_p which best describe the local statistics around a randomly chosen point p. Then, it filters edges of low weights (i.e., small values of pairwise similarities). Finally, it checks the mutual agreement for each edge. The steps of the proposed method are illustrated in Fig.2.

1. A point p in a dense cluster; the point p is depicted by ×

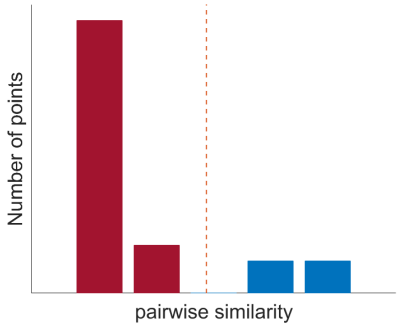


2. Use distances to neighbors to find σ_p that best describe local statistics around point p

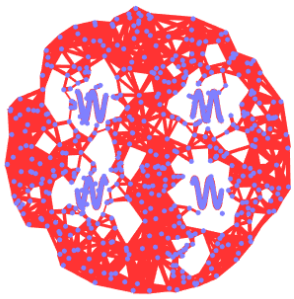


3. Use σ_p for similarity between p and q

$A_{pq} = \exp\left(\frac{-d^2(p,q)}{\sigma_p \sigma_q}\right)$ and find a threshold



4. Apply to all points to get the graph



5. Performing spectral clustering on the proposed graph

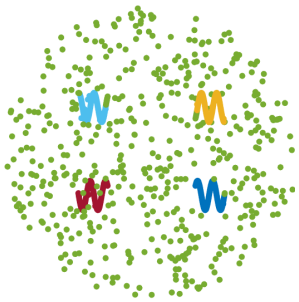


Fig. 2. Steps for the proposed approach (best viewed in color).

3.1. Finding the value of σ_p

To compute pairwise similarities (i.e., graph edges) we used the similarity measure introduce by Zelnik-Manor and Perona [14]. It is defined as follows:

$$A_{pq} = \exp\left(\frac{-d^2\left(p,q\right)}{\sigma_p \sigma_q}\right). \tag{3}$$

What is good about this similarity measure is that it uses two sources of information to compute the pairwise similarity. Points are evaluated based on: 1) distance between them, and 2) local statistics around each point. Points belonging to clusters with different statistics would have a low similarity even if they are separated by a small distance This makes this measure superior in highlighting different clusters separated by a small distance.

One problem arises from using this measure in equation 3 is how to set the value of σ_p in the denominator. In previous studies, it was set as the distance to the 7^{th} neighbor [14, 15]. However, there is no evidence that the distance to the 7^{th} 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 automatically set this parameter, we first have to identify the neighborhood around p. Then, we set σ_p to be the mean distance in this neighborhood. By looking at the histogram of distances from p in the middle column of Fig.3, we can see an early peak because p is sitting in a dense cluster. Moving to the right column in Fig.3 where p is part of a sparse cluster. We can see that the peak was pushed further along the histogram. This peak represents points in a dense cluster. It takes a

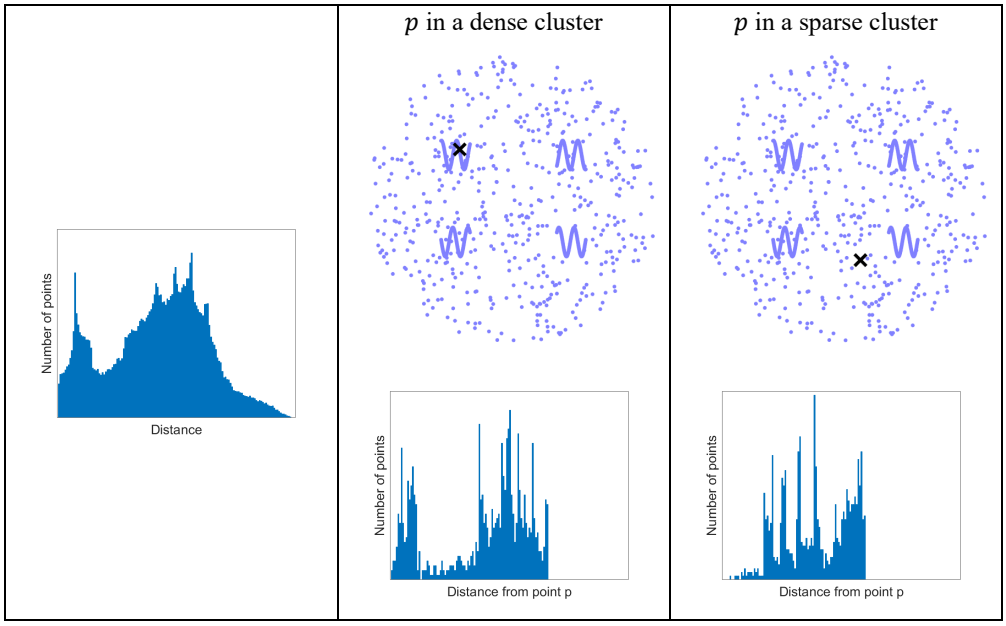


Fig. 3. Characterizing neighborhood around a point p. First column is a histogram for all distances in the input graph. Second and third columns are histograms for p (labeled as $\times$) sitting in dense and sparse clusters (best viewed in color).

small distance to reach this peak if p is in a dense cluster, but if p is in a sparse cluster it requires a long distance to reach this peak. One problem was how to set the bin width for the histogram of each point. We first compute the histogram of all distances in the input graph using Freedman-Diaconis rule [33]. The bin width in the this serves as a common ground to build a histogram for each point.

The histogram bin values of each point are smoothed using Moving Weighted Average (MWA). The smoothing was designed as follows:

$$MWA_i = \frac{v_{i-1} + v_i + v_{i+1}}{r_{i-1} + r_i + r_{i+1}}, \tag{4}$$

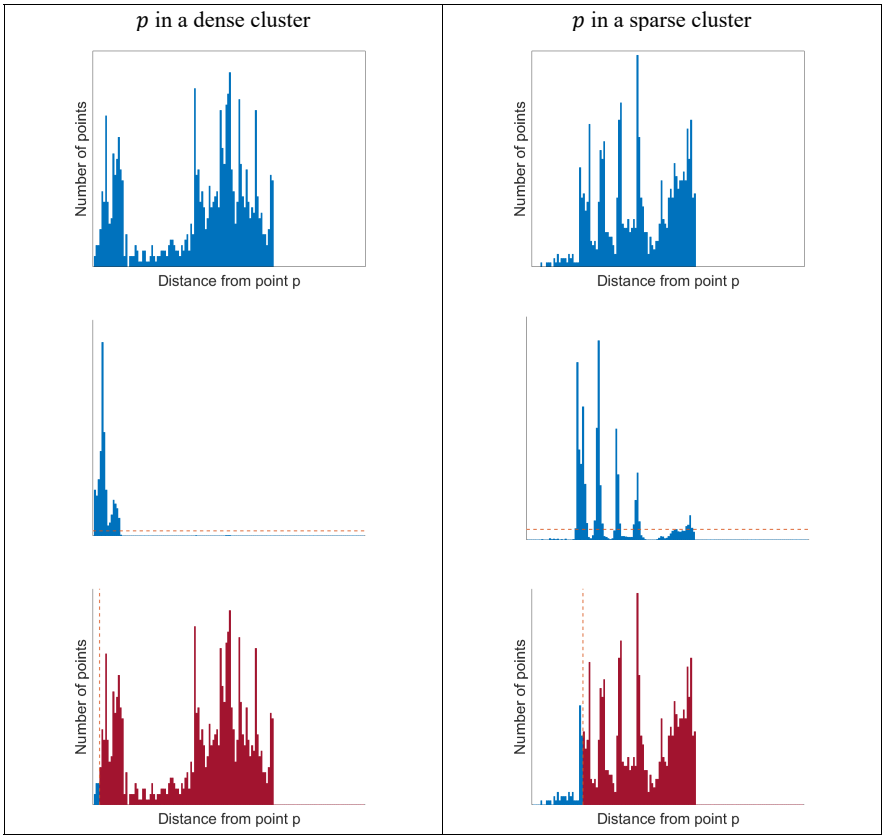


Fig. 4. The process of finding σ_p. First row is the histogram of distances form point p. Second row is the histogram bin values after applying MWA smoothing (threshold is the dashed horizontal line). Third row, σ_p is the mean of the blue bins after excluding all distances in the red bins (best viewed in color).

where v is the value of the bin and r is the rank of the bin, with $r = 1$ is the bin containing closest points to the point p. This smoothing assigns weights to bins based on distance from p. This means high weights to closer bins and low weights to further bins. We set a threshold as the mean of bin values after smoothing as shown in middle row of Fig.4. The value of σ_p is set to the mean of all distances until the first peak that is larger than the threshold. The blue bins in the third row of Fig.4 are included in σ_p, whereas the red

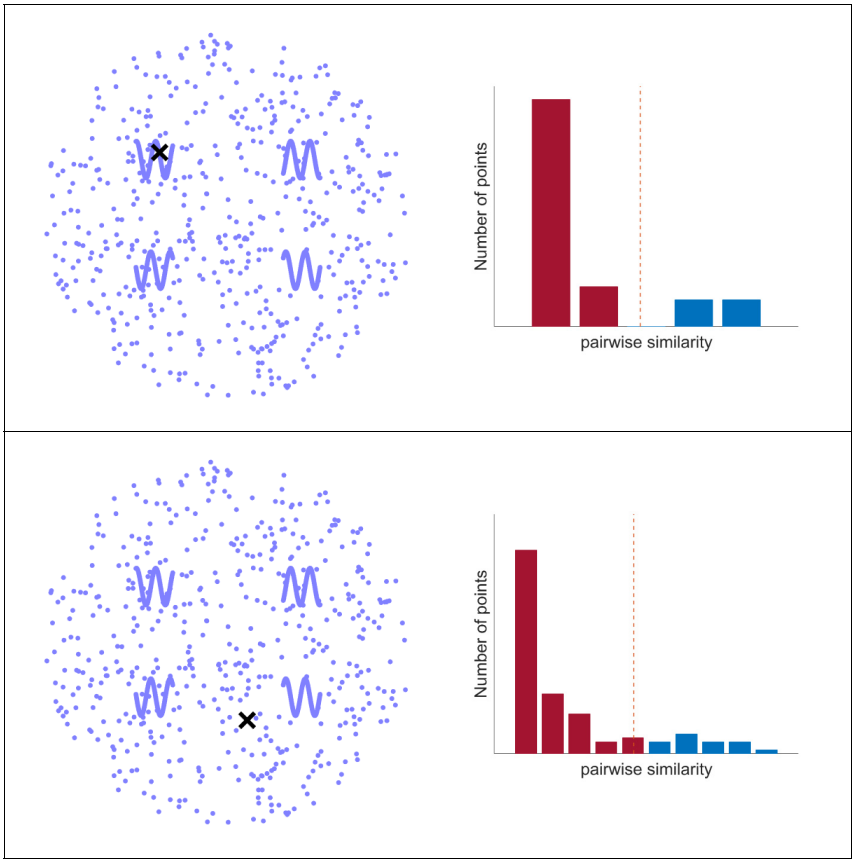


Fig. 5. Included similarities (i.e., edges) are shown in blue bins and excluded similarities in red bins for the point p in dense or sparse cluster (best viewed in color).

bins were excluded.

3.2. Computing and reducing graph edges

Once we have σ_p for each point, we can calculate the pairwise similarities using the formula in equation 4. Large values indicate highly similar points, whereas low values indicate dissimilarity. We build a histogram of all pairwise similarities using Freedman-Diaconis rule. Then, we exclude the similarities in the first bin because they are very low similarities. For each

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.
Save bin width in H_D to the variable bin_D.
3. for $p = 1$ to N — $O(Nk_{max})$
Construct a histogram H_p of $D_{p,1 \text{ to } k_{max}}$ using bin_D.
Apply MWA to bin values in H_p (equation 4).
Set v_p as the first bin that exceeds MWA threshold.
$\sigma_p = \text{mean}(D_{p,1 \text{ to } v_p})$.
4. for $p = 1$ to N — $O(Nk_{max})$
for $q = 1$ to k_{max}
$A_{p,q} = \exp\left(\frac{D(p,q)}{\sigma_p \sigma_q}\right)$
5. for $p = 1$ to N — $O(Nk_{max})$
if $\max(A_{p,1 \text{ to } k_{max}}) > \mu(A_{p,1 \text{ to } k_{max}}) + \sigma(A_{p,1 \text{ to } k_{max}})$
$A_{p,1 \text{ to } k_{max}} < \mu(A_{p,1 \text{ to } k_{max}}) + \sigma(A_{p,1 \text{ to } k_{max}}) = 0$
else
$A_{p,1 \text{ to } k_{max}} < \mu(A_{p,1 \text{ to } k_{max}}) - \sigma(A_{p,1 \text{ to } k_{max}}) = 0$
6. Construct a reduced graph using affinity matrix $A(N, k_{max})$

point, similarities lower than a threshold T_p are eliminated. If maximum similarity is larger than mean plus standard deviation $\mu + \sigma$, threshold is set as $T = \mu + \sigma$. If not, the threshold is set as $T = \mu - \sigma$. Fig.5 shows included similarities as blue bins and excluded similarities as red bins. This lead to the following definition of graph edges:

$$(p, q) \in E(G) \Leftrightarrow A_{pq} > T_p. \tag{5}$$

The last step of our reduction method is to build a mutual graph. Some graph edges are accepted by one point. In a mutual graph, the pair of points

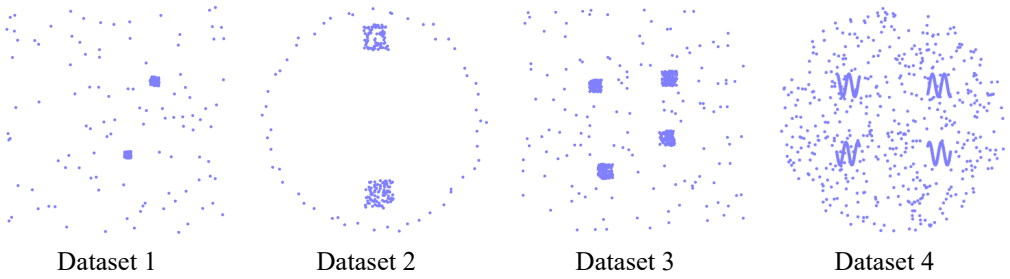


Fig. 6. Synthetic datasets used in experiments.

should agree to accept an edge. This makes the graph G to be defined as:

$$(p, q) \in E(G) \Leftrightarrow A_{pq} > T_p \quad \text{and} \quad A_{qp} > T_q. \tag{6}$$

A pseudocode illustrating the steps of the proposed method is shown in Algorithm 1. The column on the right shows the computational complexity of steps requiring higher computations.

4. Experiments and discussions

In the experiments we used four synthetic datasets shown in Fig.6. Dataset 1 to 3 were created by Zelnik-Manor and Perona [14], while Dataset 4 was created by us. We also used seven real datasets (see Table 1). Apart from MNIST dataset, all real datasets were retrieved from UCI machine learning repository [1]. Each dataset was run against two parameter sets to evaluate how methods perform under different settings of parameters.

Six methods were used for comparison and shown in Table. Methods 1 to 5 [13, 27, 28] rely on the parameter m that is the number of representatives

[1] `https://archive.ics.uci.edu/ml/index.php`

Table 1. Four Synthetic and seven real datasets used in experiments. N is number of points, d is number of dimensions, C is 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								Real datasets*							
				Parameter set 1		Parameter set 2						Parameter set 1		Parameter set 2	
	N	d	C	m	μ_0	m	μ_0		N	d	C	m	μ_0	m	μ_0
Dataset 1	303	2	3	32	7	16	3	iris	150	4	3	32	7	16	3
Dataset 2	238	2	3	32	7	16	3	wine	178	13	3	32	7	16	3
Dataset 3	622	2	5	64	7	32	3	BC-Wisc.	699	9	2	40	7	20	3
Dataset 4	1304	2	5	256	7	128	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

* apart from MNIST, all real datasets were retrieved from UCI machine learning repository (https://archive.ics.uci.edu/ml/index.php)

** after PCA reduction

to build the graph G. Therefore, these methods always performed better on the second evaluation metric $E\%$. Because they are using a graph with m vertices where $m \ll N$. Method 6 [32] relied on the parameter μ_0 to build the graph G. μ_0 is the number of neighbors where their mean was used as a threshold to include or exclude further neighbors.

All methods were evaluated using two evaluation metrics: 1) the Adjusted

Table 2. Datasets used in experiments.

	method 1	**method 2**	**method 3**	**method 4**	**method 5**	**method 6**	**Proposed method**
Reference	Yan et al. [13]	Tasdemir [27]	Tasdemir [27]	Tasdemir et al. [28]	Tasdemir et al. [28]	Alshammari et al. [32]	
Vertices reduction	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
Influential parameter	m	m	m	m	m	μ_0	N/A

Rand Index (**ARI**) [34], and 2) the percentage of edges used compared to all edges in a full graph (**E%**). ARI is a variant of the well-known measure Rand index (RI) [35]. 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 it produces zero in case of random grouping. ARI starts by building the contingency table of size $T \times L$. The contingency table contains four categories: 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 is defined as:

$$ARI(T, L) = \frac{2(n_{00}n_{11} - n_{01}n_{10})}{(n_{00} + n_{01})(n_{01} + n_{11}) + (n_{00} + n_{10})(n_{10} + n_{11})} \ . \tag{7}$$

The computational efficiency could be measured by the method's running time. However, this could be easily influenced by the type of the machine used. We chose to measure the computational efficiency by the percentage of edges compared to edges in full graph that is $N \times N$. E% is defined as:

$$E\% = \frac{E(G_{reduced})}{E(G_{full})} \ , \tag{8}$$

All experiments were coded in MATLAB 2019a and run on a windows 10 machine (3.40 GHz CPU and 8 GB of memory). The code is available on `https://github.com/mashaan14/Spectral-Clustering`.

Table 3. Results of running methods on synthetic data. ARI: mean adjusted Rand index for 50 runs ± standard deviation, and E%: the mean percentage of edges compared to edges in a full graph for 50 runs ± standard deviation. Bold values are the best scores.

		method 1	**method 2**	**method 3**	**method 4**	**method 5**	**method 6**	**Proposed method**
Dataset 1 Parameter set 1	ARI	0.452 ± 0.11	0.430 ± 0.24	0.782 ± 0.21	0.281 ± 0.22	0.694 ± 0.21	**1.000 ± 0.00**	0.957 ± 0.11
	E%	1.08 ± 0.00	0.09 ± 0.01	0.11 ± 0.01	0.09 ± 0.01	0.11 ± 0.01	**3.54 ± 0.00**	16.06 ± 0.00
Dataset 1 Parameter set 2	ARI	0.467 ± 0.12	0.364 ± 0.21	0.609 ± 0.16	0.462 ± 0.11	0.521 ± 0.09	0.884 ± 0.00	
	E%	0.26 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	1.55 ± 0.00	
Dataset 2 Parameter set 1	ARI	0.694 ± 0.01	0.834 ± 0.13	0.871 ± 0.13	0.836 ± 0.10	0.857 ± 0.11	0.948 ± 0.00	0.974 ± 0.00
	E%	1.76 ± 0.00	0.13 ± 0.01	0.13 ± 0.01	0.13 ± 0.01	0.13 ± 0.01	4.41 ± 0.00	6.32 ± 0.00
Dataset 2 Parameter set 2	ARI	0.677 ± 0.02	0.744 ± 0.06	0.717 ± 0.03	0.700 ± 0.02	0.708 ± 0.04	**1.000 ± 0.00**	
	E%	0.43 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	**1.94 ± 0.00**	
Dataset 3 Parameter set 1	ARI	0.675 ± 0.07	0.612 ± 0.27	0.841 ± 0.10	0.544 ± 0.14	0.810 ± 0.14	0.957 ± 0.04	**1.000 ± 0.00**
	E%	1.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	1.72 ± 0.00	**1.45 ± 0.00**
Dataset 3 Parameter set 2	ARI	0.670 ± 0.03	0.804 ± 0.09	0.858 ± 0.09	0.627 ± 0.09	0.701 ± 0.05	0.848 ± 0.08	
	E%	0.26 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.76 ± 0.00	
Dataset 4 Parameter set 1	ARI	0.393 ± 0.02	0.103 ± 0.16	0.282 ± 0.20	0.061 ± 0.13	0.171 ± 0.19	0.422 ± 0.01	0.700 ± 0.13
	E%	3.84 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.72 ± 0.00	0.51 ± 0.00
Dataset 4 Parameter set 2	ARI	0.400 ± 0.01	0.466 ± 0.06	0.450 ± 0.04	0.400 ± 0.03	0.400 ± 0.02	**0.961 ± 0.00**	
	E%	0.96 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	**0.33 ± 0.00**	

4.1. Synthetic data

In synthetic datasets the proposed method delivered a performance that ranked it as 2nd, 2nd, 1st, and 2nd for datsets 1 to 4 respectively (see Table 3). Method 6 was 1st performer in three occasions. However, its performance drops significantly when we changed the parameter μ_0. For example, its performance drops by 50% in dataset 4 when we changed $\mu_0 = 3$ to $\mu_0 = 7$. In terms of used edges, the proposed method used 6.32%, 1.45%, and 0.51% of 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 full graph edges. This sharp increase could by explained that points in dense clusters were fully connected.

4.2. Real data

For datasets with small number of points N (iris, wine, and BC-Wisconsin), the proposed method delivered a performance that ranked it as 2^{nd}, 1^{st}, and 1^{st} performer respectively (shown in Table 4). The best performer in iris dataset was method 3 with $m = 16$. However, with $m = 32$ its performance drops by more than 15%. Moving to wine dataset where the second performer was method 6 with $\mu_0 = 7$. But, by setting $\mu_0 = 3$, the performance dropped by 50%. These observations show how wrong selection of parameters could severely impact the accuracy of clustering. On the other hand, the 2^{nd} performer in BC-Wisconsin dataset was not impacted by changing the parameter m from 40 to 20. The only thing that changed was the number of edges in the graph.

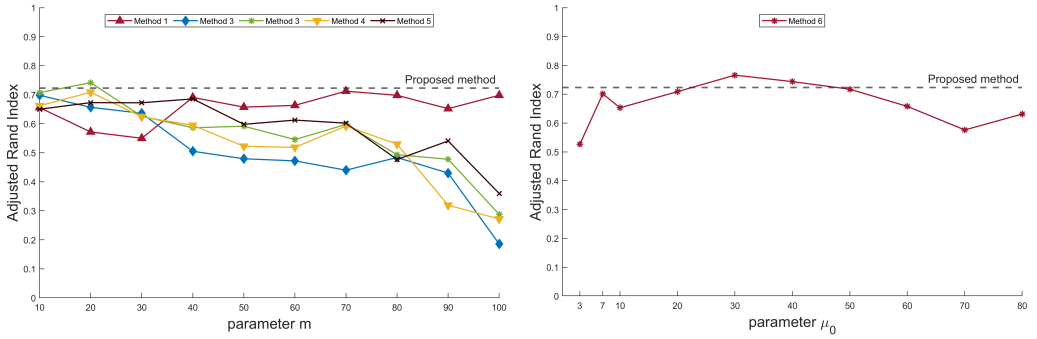
For datasets with large number of points N (statlog, MNIST, PenDigits, and mGamma) the proposed method came 7^{th}, 4^{th}, 2^{nd}, and 3^{rd}. In statlog and MNIST the proposed method did not perform well. Also, all methods using local σ (method 1 and 6) delivered poor performance. This indicates that 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. We should use CONN to discover clusters discontinuity instead of tracking local statistics. It is worth mentioning that best performers did not keep their scores when we changed the parameters. In PenDigits dataset, the proposed method was not far behind the best performer, which did not keep its performance when we changed the parameters. In mGamma dataset, method 1 delivered the best performance which was not affected by the change of parameters.

Table 4. Results of running methods on real data. ARI: mean adjusted Rand index for 50 runs ± standard deviation, and E%: the mean percentage of edges compared to edges in a full graph for 50 runs ± standard deviation. Bold values are the best scores.

		method 1	**method 2**	**method 3**	**method 4**	**method 5**	**method 6**	**Proposed method**
iris Parameter set 1	ARI	0.564 ± 0.08	0.603 ± 0.15	0.601 ± 0.10	0.621 ± 0.10	0.648 ± 0.12	0.701 ± 0.00	0.723 ± 0.00
	E%	4.44 ± 0.00	0.40 ± 0.03	0.42 ± 0.03	0.39 ± 0.02	0.42 ± 0.02	8.26 ± 0.00	4.45 ± 0.00
iris Parameter set 2	ARI	0.591 ± 0.05	0.633 ± 0.12	**0.752 ± 0.12**	0.694 ± 0.09	0.693 ± 0.10	0.527 ± 0.00	
	E%	1.07 ± 0.00	0.18 ± 0.01	0.17 ± 0.01	0.19 ± 0.02	0.17 ± 0.01	3.84 ± 0.00	
wine Parameter set 1	ARI	0.582 ± 0.10	0.337 ± 0.18	0.327 ± 0.16	0.488 ± 0.15	0.415 ± 0.19	0.914 ± 0.00	**0.930 ± 0.01**
	E%	3.15 ± 0.00	0.22 ± 0.01	0.25 ± 0.01	0.22 ± 0.01	0.25 ± 0.01	11.92 ± 0.00	7.83 ± 0.00
wine Parameter set 2	ARI	0.649 ± 0.11	0.453 ± 0.18	0.484 ± 0.17	0.661 ± 0.13	0.675 ± 0.11	0.458 ± 0.07	
	E%	0.76 ± 0.00	0.10 ± 0.01	0.11 ± 0.01	0.11 ± 0.01	0.11 ± 0.01	5.96 ± 0.00	
BC-Wisconsin Parameter set 1	ARI	0.849 ± 0.02	0.785 ± 0.12	0.858 ± 0.02	0.844 ± 0.02	0.860 ± 0.01	0.097 ± 0.15	**0.872 ± 0.00**
	E%	0.32 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	1.28 ± 0.00	3.48 ± 0.00
BC-Wisconsin Parameter set 2	ARI	0.805 ± 0.07	0.856 ± 0.02	0.852 ± 0.02	0.849 ± 0.02	0.860 ± 0.01	-0.004 ± 0.00	
	E%	0.08 ± 0.00	0.03 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.02 ± 0.00	0.64 ± 0.00	
Statlog Parameter set 1	ARI	0.441 ± 0.06	0.502 ± 0.04	0.504 ± 0.02	0.458 ± 0.04	**0.519 ± 0.02**	0.368 ± 0.02	0.470 ± 0.01
	E%	0.02 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.14 ± 0.00	0.28 ± 0.00
Statlog Parameter set 2	ARI	0.399 ± 0.09	0.492 ± 0.04	0.517 ± 0.03	0.453 ± 0.04	0.483 ± 0.02	-0.001 ± 0.00	
	E%	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.00	
MNIST Parameter set 1	ARI	0.395 ± 0.02	0.475 ± 0.07	**0.484 ± 0.04**	0.457 ± 0.02	0.450 ± 0.07	0.395 ± 0.07	0.453 ± 0.05
	E%	0.25 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.42 ± 0.00	1.07 ± 0.00
MNIST Parameter set 2	ARI	0.367 ± 0.03	0.428 ± 0.03	0.429 ± 0.04	0.372 ± 0.03	0.372 ± 0.02	0.443 ± 0.08	
	E%	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.21 ± 0.00	
PenDigits Parameter set 1	ARI	0.510 ± 0.06	0.578 ± 0.23	**0.695 ± 0.06**	0.412 ± 0.28	0.391 ± 0.35	0.000 ± 0.00	0.671 ± 0.05
	E%	0.21 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.17 ± 0.00	0.52 ± 0.00
PenDigits Parameter set 2	ARI	0.512 ± 0.07	0.637 ± 0.07	0.619 ± 0.07	0.544 ± 0.05	0.563 ± 0.05	0.000 ± 0.00	
	E%	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.09 ± 0.00	
mGamma Parameter set 1	ARI	**0.080 ± 0.00**	0.017 ± 0.03	0.033 ± 0.03	0.034 ± 0.04	0.027 ± 0.03	0.056 ± 0.02	0.059 ± 0.00
	E%	0.28 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.10 ± 0.01	0.41 ± 0.00
mGamma Parameter set 2	ARI	0.080 ± 0.00	0.019 ± 0.03	0.042 ± 0.03	0.042 ± 0.04	0.023 ± 0.03	0.000 ± 0.00	
	E%	0.28 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.01	

4.3. The effect of parameters on spectral clustering performance

In this experiment, we investigate how a wide selection of parameters could affect the accuracy of spectral clustering. For this we used iris dataset which is a well-known dataset with relatively small number of points N to limit the range of parameters selection. The parameters m and μ_0 were given the following values: $m \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and $\mu_0 \in \{3, 7, 10, 20, 30, 40, 50, 60, 70, 80\}$.



(a) different values of m **(b)** different values of μ_0

Fig. 7. Testing the methods performance on iris dataset under different settings of parameters m and μ_0 (best viewed in color).

In Fig. 7 (left), methods 1 to 5 were run with different values of m. The performance of these methods fluctuates with different values of m, with a clear downward trend as m increased. The dashed horizontal line is the performance of the proposed method. It provides best accuracy and stable alternative since it does not rely on the parameter m. In Fig. 7 (right), method 6 was run with different values of μ_0. It starts with low performance and peaked around $\mu_0 = 30$, then it took a downward trend. By eliminating the use of μ_0, our method delivered a stable performance showed by the horizontal dashed line.

5. Conclusion

The problem of detecting non-convex clusters has led to the development of numerous clustering methods. These methods can be classified into: kernel-based and graph-based methods. The difficulty of determining a good kernel that provides a good separation of clusters has made kernel-based methods less attractive. One of the well-known graph-based methods is

spectral clustering which could detect clusters of any shape. It uses eigen-decomposition to map vertices on the graph into an embedding space where clusters can be separated by k-means. Despite the efficiency of this step, it demands high computations of $O(n^3)$. Therefore, multiple studies have developed algorithms to reduce the graph size. The goal of reducing the graph size has led to an 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 maintained spectral clustering accuracy. It takes an input as a full graph or a k-nearest neighbor graph (in case of large number of points). Then, it reduces the graph edges using statistical measures that require low computations. The experiments reveal 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. This is because most of vertices reduction methods available in the literature overlook local statistics which could affect spectral clustering accuracy. A useful extension of the proposed method would be a vertices reduction component that is aware of local statistics.

References

[1] T. Kohonen, Self-organizing maps, Springer series in information sciences ; 30, 3rd ed., Springer, Berlin ;, 2001.

[2] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory 28 (1982) 129–137. doi:10.1109/TIT.1982.1056489.

[3] G. J. McLachlan, D. Peel, Finite mixture models, Wiley series in probability and statistics Applied probability and statistics section, Wiley, New York, 2000.

[4] J. Liu, J. Han, Spectral Clustering, Taylor & Francis, 2013, pp. 177–200.

[5] R. Vidal, Y. Ma, S. Sastry, Generalized Principal Component Analysis, Springer New York, 2016.

[6] Y. Qin, Z. L. Yu, C.-D. Wang, Z. Gu, Y. Li, A novel clustering method based on hybrid k-nearest-neighbor graph, Pattern Recognition 74 (2018) 1–14. URL: http://www.sciencedirect.com/science/article/pii/S0031320317303497. doi:https://doi.org/10.1016/j.patcog.2017.09.008.

[7] Y. Kim, H. Do, S. B. Kim, Outer-points shaver: Robust graph-based clustering via node cutting, Pattern Recognition 97 (2020) 107001. URL: http://www.sciencedirect.com/science/article/pii/S0031320319303048. doi:https://doi.org/10.1016/j.patcog.2019.107001.

[8] H. Li, X. Liu, T. Li, R. Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognition 102 (2020) 107206. URL: http://www.sciencedirect.com/science/article/pii/S0031320320300121. doi:https://doi.org/10.1016/j.patcog.2020.107206.

[9] I. Tyuryukanov, M. Popov, M. v. d. Meijden, V. Terzija, Discovering clusters in power networks from orthogonal structure of spectral embedding, IEEE Transactions on Power Systems (2018). doi:10.1109/TPWRS.2018.2854962.

[10] R. Zhang, F. Nie, X. Li, Self-weighted spectral clustering with parameter-free constraint, Neurocomputing 241 (2017) 164–170. URL: http://www.sciencedirect.com/science/article/pii/S0925231217303089. doi:https://doi.org/10.1016/j.neucom.2017.01.085.

[11] W. Jiang, W. Liu, F.-l. Chung, Knowledge transfer for spectral clustering, Pattern Recognition 81 (2018) 484–496. URL: http://www.

sciencedirect.com/science/article/pii/S0031320318301511. doi:https://doi.org/10.1016/j.patcog.2018.04.018.

[12] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in Neural Information Processing Systems (2002).

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

[14] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, Advances in Neural Information Processing Systems (2005) 1601–1608.

[15] M. Sugiyama, Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis, Journal of machine learning research 8 (2007) 1027–1061.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining 96 (1996) 226–231.

[17] R. Xu, D. Wunsch, Clustering, Wiley-IEEE Press, 2009. doi:10.1002/9780470382776.

[18] L. Ertöz, M. Steinbach, V. Kumar, Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data, 2003, pp. 47–58. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611972733.5. doi:10.1137/1.9781611972733.5.

[19] R. A. Jarvis, E. A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Transactions on Computers C-22 (1973) 1025–1034. doi:10.1109/T-C.1973.223640.

[20] T. Inkaya, S. Kayalıgil, N. E. Ozdemirel, An adaptive neighbourhood construction algorithm based on density and connectivity, Pattern Recognition Letters 52 (2015) 17–24. URL: http://www.sciencedirect.com/science/article/pii/S0167865514002815. doi:https://doi.org/10.1016/j.patrec.2014.09.007.

[21] T. Inkaya, A parameter-free similarity graph for spectral clustering, Expert Systems with Applications 42 (2015) 9489–9498. URL: http://www.sciencedirect.com/science/article/pii/S0957417415005345. doi:https://doi.org/10.1016/j.eswa.2015.07.074.

[22] J. Shi, J. Malik, Normalized cuts and image segmentation, Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97) (1997) 731–737. doi:Doi10.1109/Cvpr.1997.609407.

[23] Y. Weiss, Segmentation using eigenvectors: A unifying view, Proceedings of the IEEE International Conference on Computer Vision 2 (1999) 975–982. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0033285447&partnerID=40&md5=1378b78305944f8f4fde7c0650ea46d1.

[24] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (2007) 395–416. doi:10.1007/s11222-007-9033-z.

[25] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms 07-09-January-2007 (2007) 1027–1035.

[26] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (1990) 1464–1480.

[27] K. Tasdemir, Vector quantization based approximate spectral clustering of large datasets, Pattern Recognition 45 (2012) 3034–3044. doi:https://doi.org/10.1016/j.patcog.2012.02.012.

[28] K. Tasdemir, B. Yalcin, I. Yildirim, Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures, Pattern Recognition 48 (2015) 1465–1477. doi:https://doi.org/10.1016/j.patcog.2014.10.023.

[29] M. Alshammari, M. Takatsuka, Approximate spectral clustering using topology preserving methods and local scaling, The 25th International Conference on Neural Information Processing (ICONIP 2018) (2018).

[30] M. Alshammari, M. Takatsuka, Approximate spectral clustering density-based similarity for noisy datasets, Pattern Recognition Letters

128 (2019) 155–161. URL: http://www.sciencedirect.com/science/article/pii/S0167865518305439. doi:https://doi.org/10.1016/j.patrec.2019.08.020.

[31] C. D. Correa, P. Lindstrom, Locally-scaled spectral clustering using empty region graphs, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2012) 1330–1338. doi:10.1145/2339530.2339736.

[32] M. Alshammari, J. Stavrakakis, M. Takatsuka, Refining a k-nearest neighbor graph for a computationally efficient spectral clustering, submitted to Pattern Recognition Journal (2020).

[33] D. Freedman, P. Diaconis, On the histogram as a density estimator:l2 theory, Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete 57 (1981) 453–476. URL: https://doi.org/10.1007/BF01025868. doi:10.1007/bf01025868.

[34] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1985) 193–218. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0000008146&doi=10.1007%2fBF01908075&partnerID=40&md5=bd03cf70caee7de0ccf3c0dd431b97ca. doi:10.1007/BF01908075.

[35] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66 (1971) 846–850. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84950632109&doi=10.1080%2f01621459.1971.10482356&partnerID=40&md5=3bcb39c5cbd4ccf7dec3e0fa080b1759. doi:10.1080/01621459.1971.10482356.