Approximate Spectral Clustering Using Topology Preserving Methods and Local Scaling

Mashaan Alshammari(✉) and Masahiro Takatsuka

School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia
mals6571@uni.sydney.edu.au,
masa.takatsuka@sydney.edu.au

Abstract. Spectral clustering is the type of unsupervised learning that separates data based on their connectivity instead of convexity. However, its computational demands increase cubically with the number of points n. This triggered a stream of studies to ease these demands. An effective solution is to provide an approximated graph $G^* = (V^*, E^*)$ for the input data with a reduced set of vertices and edges. Recent similarity measures used to construct the approximated graph $G^* = (V^*, E^*)$ have some deficiencies such as: (1) weights on edges highly depend on the cluster density, and (2) larger memory footprint compared to conventional similarity measures. In this work, we employed topology preserving methods (e.g., neural gas) to obtain $G^* = (V^*, E^*)$ due to their ability to preserve input data topology. Then we used a conventional similarity measure to assign weights on the graph. The experiments reveal that graphs obtained through topology preserving methods and passed to a locally scaled similarity measure, produce performances comparable to the recent measures with a significantly smaller memory footprint.

Keywords: Spectral clustering · Topology preserving methods
Neural gas

1 Introduction

Spectral clustering uses the spectrum of the affinity matrix A to partition the connected components of the graph G that connects data points $\{x_1, x_2, \ldots, x_n\}$. Unfortunately, storing A requires $O(n^2)$ and performing eigendecomposition needs computations of $O(n^3)$ [1]. Due to the effectiveness of spectral clustering, many studies investigated minimizing its computational demands. Reducing the graph vertices V shrinks the size of A from $n \times n$ to $m \times m$, where $m \ll n$. On the other hand, reducing the number of edges E results in a sparse matrix [2] which reduces the overall computations. Nevertheless, achieving both reductions is not trivial and has been the subject for most of spectral clustering researchers [2].

The idea behind using m representatives out of n points is to perform spectral clustering on a reduced set of points then generalize the outcome. The selection of representatives could be obtained through sampling [2]. The more sophisticated the sampling scheme, the more we are confident that small clusters are not left out. A more effective approach would be using a learning algorithm to place the representatives [1]. However, both approaches are incapable of constructing the graph $G(V, E)$ and we have to rely on the similarity measure to produce a sparse graph.

To construct the graph $G(V, E)$ we could use a similarity measure penalized by a global scale (σ) which controls the decay of the affinity as proposed in [3]. Nevertheless, σ is insensitive to local statistics [4] and difficult to tune. An alternative solution proposed in [4] is to use a local scale σ_i ($i \in \{1, 2, \ldots, m\}$) set as the distance to K^{th} neighbor. However, K is another parameter that needs tuning. Moreover, local scaling measure is unable to produce a sparse graph (i.e., eliminating some edges) and it is highly depending on the graph selection like *knn* or ε graphs. Recently, novel similarity measures were introduced [2]. On one hand, these measures can produce a sparse graph by eliminating weak edges. On the other hand, their computational demands are much higher than their conventional counterparts.

In this work, we propose the use of topology preserving methods and demonstrate that they are more efficient in placing representatives to approximate input data. Unlike other approximation methods such as KASP [1], topology preserving methods can produce a sparse graph, since edge drawing is part of their training. A topology preserving method has two components: vector quantization and establishing lateral connections [5]. We tested self-organizing maps (SOM) [6], neural gas (NG) [7], and growing neural gas (GNG) [8]. We found out that graphs obtained through topology preserving methods improve the accuracy of the local scaling measure with markedly smaller memory footprint than other measures.

2 Related Work

Two topics were the subject for most research on spectral clustering. The former relates to the question: how to draw edges between data points by quantifying their similarity. The second topic is approximate spectral clustering (ASC) which concerns of placing m representatives to cluster n points.

2.1 Similarity Measures for Spectral Clustering

Defining similarities between data points is crucial for spectral clustering accuracy. Those similarities are represented as weights on edges of the graph G which should be informative to highlight the optimal cut for spectral clustering. Ideally, large weights are assigned between the points in same cluster, and relatively small weights (or no edges at all) between points in different clusters. There are number of options for the similarity measures (see Appendix D in [9]). Globally scaled similarity which is the Euclidean distance between the compared points penalized by a Gaussian scale:

$$A_{ij} = \exp\left(\frac{-d^2(i,j)}{\sigma^2}\right) \tag{1}$$

However, this similarity measure ignores the local statistics around the compared points [4]. An advanced option introduced in [4], that penalizes the Euclidean distance by the product of local scales σ_i. and σ_j. The local scale of a particular point is set as the distance to its K^{th} neighbor:

$$A_{ij} = \exp\left(\frac{-d^2(i,j)}{\sigma_i \sigma_j}\right) \tag{2}$$

The local scale set as $\sigma_i = d(i, i_K)$. This measure is sensitive to local statistics and adjusts itself accordingly. However, setting the K^{th} neighbor is an issue for this measure. All aforementioned measures are unable to eliminate edges to produce a sparse affinity matrix. To avoid this, one could restrict the Euclidian distance to a certain threshold [9]. Unfortunately, this means one more parameter that needs tuning.

There are number of similarity measures that can produce sparse versions of the graph G. A measure called common-near-neighbor (CNN) is defined as the number of points in the intersection of two spheres with radius ε centered at i and j [10]. The weight of the edge in CNN graph is determined by the number of shared points. Intuitively, if there are no shared points between i and j, there will be no edge connecting them, hence it results in a sparse graph. Nevertheless, in large datasets computing CNN requires heavy computations [2]. Connectivity matrix (CONN) is suitable for vector quantization methods [2]. It utilizes the concept of induced Delaunay triangulation presented in [5]. $CONN(i,j)$ is defined as the number of points which i and j are their best-matching-unit and second-best-matching-unit. Like CNN, CONN can eliminate edges in case of no common neighbors. However, CONN is much faster than CNN since its computations are embedded in vector quantization training.

$$CONN(i,j) = \left|\left\{v \in V_{ij} \cup V_{ji}\right\}\right| \tag{3}$$

2.2 Approximate Spectral Clustering

High computational demands of spectral clustering stimulate a stream of studies addressing what is known as approximate spectral clustering (ASC) [1, 2]. The basic idea of ASC is to carry out the spectral clustering with m representatives, where $m \ll n$, then generalize the results to the entire dataset. However, the fundamental question is how to choose m representatives. Choosing m via random sampling, introduces the risk of missing small clusters. A different approach is to set a low rank approximation of the affinity matrix A (e.g., Nystrom method). However, some studies reported high memory consumption of such methods [1].

Vector quantization methods (e.g., k. -means and SOM) proved to be an efficient way to approximate spectral clustering [1, 2]. Even though they require a training time, the m representatives inherent density information found in n points. Thus, the training step mitigates the risk of missing small clusters. The initial effort in this direction was by Yan et al. [1], where they used k-means to approximate n points. However, k-means is unable to preserve the topology of the input data. Topology preserving methods, namely self-organizing maps (SOM) and neural gas (NG), were used to approximate input data [2]. Nevertheless, the edges drawn by SOM and NG were replaced by the edges drawn via CONN [2]. In other words, the vector quantization component of SOM and NG was used, whereas the lateral connection component was not utilized. This means the training time spent on lateral connections was wasted.

The approximation process should place representatives and draw edges between them to justify its training time. If we could get edges through the approximation step, the similarity measure will weigh those edges and skip non-existing ones. This will result in a sparse affinity matrix A. Topology preserving methods are perfectly suited to this objective, since they produce a new graph with a reduced set of representatives.

3 Proposed Approach

The algorithm passes through four main steps to perform approximate spectral clustering with smaller memory footprint. (1) It approximates the input data using one of four methods: k-means, SOM, NG, or GNG, each of which is coupled with an edge drawing scheme. (2) The weights on edges were set using local σ [4] or *CONN* [2] followed by an eigendecomposition for the graph Laplacian L. (3) It passes through a cost function that automatically selects the dimensions needed to construct an embedding space where the clusters could be detected. (4) The number of clusters in the embedding space was estimated by another cost function.

3.1 Approximating Input Data

The computational bottleneck in the spectral clustering could be avoided by performing vector quantization and carry on with a reduced number of representatives. For this purpose, four vector quantization methods were used to produce an approximated graph, three of which contain a topology preserving component. A map M satisfies the topology preserving condition if points i and j which are adjacent in $\mathbb{R}^d$ are mapped to neurons w_i and w_j which are adjacent in M, and, vice versa [11].

k-means. k-means is a well-known method for vector quantization. Given data points $\{x_1, x_2, \ldots, x_n\}$ distributed around m centroids $\{\mu_1, \mu_2, \ldots, \mu_m\}$. It attempts to minimize the squared distances between data points and their closest centroids:

$$\min_{\{\mu_i\},\{c_{ij}\}} \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} ||x_i - \mu_i||^2 \tag{4}$$

where $c_{ij} \in \{0, 1\}$ such that $c_{ij} = 1$ if x_i was assigned to the cluster μ_j and $c_{ij} = 0$ otherwise. k-means was used to approximate input data. Unlike SOM, k-means is not equipped with a topology preserving component. Therefore, its centroids were connected using nearest neighbor graphs (NN).

Self-Organizing Map. SOM consists of set of neurons attached to lateral connections. During SOM training, the map attempts to capture input data topology. Its training starts by randomly selecting a data sample x_i. Its closest neuron is called the best matching unit w_b:

$$||x_i - w_b|| = \min_j\{||x_i - w_j||\} \tag{5}$$

This process is the competitive stage in which neurons compete to win x_i. Then, w_b pulls its topological neighbors to be closer in a process known as the cooperative stage:

$$w_j(t+1) = w_j(t) + \alpha(t)\eta(t)\left(x_i - w_j(t)\right) \tag{6}$$

where $\alpha(t)$ is the learning rate monotonically decreasing with time t and $\eta(t)$ is the neighborhood kernel. In our case, SOM was set to 2D hexagonal grid.

Neural Gas. One of SOM deficiencies is that the network adaption is performed based on grid connections regardless of neurons positions in the input space. Another problem is the fixed connections which may restrict its ability to capture data topology. These two problems were addressed by the neural gas (NG) [11]. NG initiates neurons in the input space without lateral connections. Initially, a random point x_i is introduced to neurons and its best matching unit w_b is identified. Other neurons are ordered ascendingly based on their distance from w_b and updated as per the following adaption rule:

$$w_j(t+1) = w_j(t) + \varepsilon \cdot e^{-k_j/\lambda}\left(x_i - w_j(t)\right) \tag{7}$$

where $j \in \{1, 2, \ldots, m\}$, k is the rank associated with w_j based on its distance from w_b. $\varepsilon \in \{0, 1\}$ controls the extent of the adaption and λ is a decaying constant. Next, NG applies the competitive Hebb learning (CHL) to connect best matching unit w_b^0 and the second best matching unit w_b^1, to maintain a perfectly topology preserving map (see Theorem 2 in [11] and the discussion therein). Due to neurons movement, these edges are allowed to age and perhaps removed if they are not refreshed. In our experiments, we set the stopping criteria to the stability in quantization error.

Growing Neural Gas. Although neural gas was an improvement over SOM, it kept some properties of original SOM. It uses a fixed number of neurons and relies on decaying parameters for adaption. These two deficiencies were overcome by growing neural gas (GNG). It starts by introducing a random x_i to the competing neurons and selects the best matching unit and computes its error using:

$$error_{t+1} = error_t + ||w_b - x_i||^2 \tag{8}$$

w_b and its topological neighbors are adapted using the following adaption rules:

$$w_b(t+1) = \varepsilon_b(x_i - w_b) \tag{9}$$

$$w_k(t+1) = \varepsilon_k(x_i - w_k) \tag{10}$$

where k indicates all direct topological neighbors of w_b. ε_b and ε_k are the fractions of change. Then, GNG applies competitive Hebb learning (CHL) to connect BMU and the second BMU and set the weight on the edge to zero. If the current iteration is an integer multiple of the parameter l, a new neuron is inserted in halfway between the neuron with the maximum accumulated error w_q and its topological neighbor with the largest error w_f. The newly inserted neuron is connected to both w_q and w_f, and the old edge connecting them is removed. The training continues until the stopping criteria is met, which was set in our experiments to the stability in quantization error.

3.2 Constructing the Affinity Matrix A

To construct the affinity matrix A, a similarity score should be set for each pair of neurons. We used two measures to set the similarity between neurons. The first is local σ [4] defined as:

$$A_{w_i w_j} = \exp\left(\frac{-d^2(w_i, w_j)}{\sigma_{w_i}\sigma_{w_j}}\right) \tag{11}$$

It cannot draw a sparse graph by itself, because its formula does not produce zero values. Therefore, we have to rely on edges in the graph obtained through approximation methods. This similarity measure can preserve the local statistics between the neuron w_i and w_j regardless of the density around them as illustrated in Fig. 1. It is obvious that the density of the points on the outer ring is much less than the middle ring. However, this similarity measure set similar weights on the outer ring edges as the middle ring.

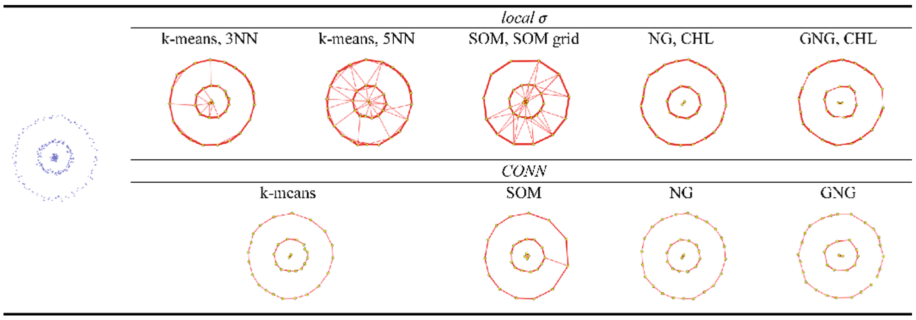

Fig. 1. Constructing an approximated graph $G^* = (V^*, E^*)$ using local σ and *CONN*

The second measure is *CONN* [2]. It is suitable for vector quantization since it measures the density between neurons. Unlike local σ, the weight of the edge is entirely determined by the number of points in the Voronoi region $V_{w_i w_j}$ shared by w_i and w_j. Examples of *CONN* graphs shown in Fig. 1

$$A_{w_i w_j} = \left| \left\{ \mathrm{v} \in V_{w_i w_j} \cup V_{w_j w_i} \right\} \right| \tag{12}$$

Setting K for Computing Local σ. Selection of the K^{th} neighbor is a problem for the use of local σ. Although some studies have used $K = 7$ [4, 9], it remains as an empirical selection and have not been used in approximate spectral clustering. Performing vector quantization affects our decision to set K. It is probable that the selected value has no actual edge in the approximated graph $G^* = (V^*, E^*)$.

Let K be the direct neighbor of the current neuron w_i (i.e., $K = 1$). For nearest neighbor graphs, this edge is guaranteed to exist since each neuron has k edges (i.e., 3 or 5 edges). For SOM with a hexagonal grid the existence of this edge is guaranteed since each neuron on that grid has at least 2 edges. For NG and GNG, the existence of this edge is guaranteed by the definition of competitive Hebbian rule that connects best matching neuron with the second best matching neuron. Therefore in all experiments we set $K = 1$ that is the direct neighboring neuron.

Memory Footprint for Local σ and *CONN*. Computing each of the similarity measures, requires a certain array needs to be present in the memory. That array contains values needed to compute the similarities between neurons. Starting with local σ, it requires a value attached to each neuron. That value represents the local scale σ_i. Therefore, a one-dimensional array of size $1 \times m$ is needed to be present in the memory to construct the affinity matrix A using the local scaling measure. For *CONN* similarity measure, the edge between w_i and w_j is determined by the number of points where w_i is the BMU and w_j is the second BMU, in addition to the points where w_j is the BMU and w_i is the second BMU. To achieve that, a two dimensional array is needed.

3.3 Embedding Space Dimensions $\mathbb{R}^{m \times k}$

The graph Laplacian is computed as $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. Then, eigendecompsition is performed on L to obtain the largest eigenvectors that partition the graph. In the original spectral clustering algorithm in [3], it is recommended to build an embedding space using k eigenvectors then run k-means on that space to find clusters. In that work, k was set manually, however, automating the selection of k would make the algorithm more practical. The automation could be achieved by counting the number of eigenvalues of multiplicity zero. However, eigenvalues could deviate from zero due to noise leaving this technique unreliable [4].

For an eigenvector v_i to be included in the embedding space $\mathbb{R}^{\boldsymbol{m \times k}}$, it must be able to sperate the neurons. An eigenvector that cannot separate the neurons could confuse the clustering in the embedding space. v_i discrimination power could be measured by: (1) a clustering index that gives a score on how well data is separated, and (2) its corresponding eigenvalue λ_i. The closer λ_i to zero the more discriminative v_i becomes. Therefore, we used the following formula to evaluate all eigenvectors $\{v_1, v_2, \ldots, v_m\}$:

$$\frac{\sum_{c=2}^{4} DBI_c(v_i)}{\lambda_i}, \qquad 2 \le i \le m \tag{13}$$

For every eigenvector v_i containing one dimensional data $\{w_1, w_2, \ldots, w_m\}$, we compute how well it can separate them into 2, 3, and 4 clusters ($c \in \{2, 3, 4\}$) using the Davies-Bouldin index (DBI). This quantity was penalized by the eigenvalue λ_i corresponding to the eigenvector v_i The Davies-Bouldin index is defined as:

$$\frac{1}{c}\sum_{i=1}^{c} \max_{i \neq j} \left\{ \frac{S_c(Q_i) + S_c(Q_j)}{d_{ce}(Q_i, Q_j)} \right\} \tag{14}$$

Given neurons $\{w_1, w_2, \ldots, w_m\}$ clustered into $\{Q_1, Q_2, \ldots, Q_c\}$ clusters. $S_c(Q_i)$ is within-cluster distances in cluster i, and $d_{ce}(Q_i, Q_j)$ is the distance between clusters i and j. After scoring all eigenvectors $\{v_1, v_2, \ldots, v_m\}$, the scores were fit into a histogram. The bin size was determined via Freedman-Diaconis rule, defined as $2Rm^{-1/3}$, where R is the inter-quartile range. The desired eigenvectors fall outside the interval $[\mu \pm \sigma]$, where μ is the mean and σ is the standard deviation.

3.4 Number of Clusters in the Embedding Space

The embedding space $\mathbb{R}^{m \times k}$ is spanned by eigenvectors qualified through the method explained in the previous subsection. In this space the connected components of the graph $G^* = (V^*, E^*)$ form convex clusters could be detected by k-means. However, the parameter k must be set prior to performing k-means.

One way to select k would be through a clustering index (like DBI in Eq. 14) then use k that yields the lowest score. However, this approach tends to favor large values of k for better separation. This could be avoided by examining the eigenvalues of the graph Laplacian L. As stated in [12] the number of connected components of the graph equals the number of eigenvalues with multiplicity zero. Therefore, we should penalize each value of k by the sum of eigenvalues it accumulates (Fig. 2).

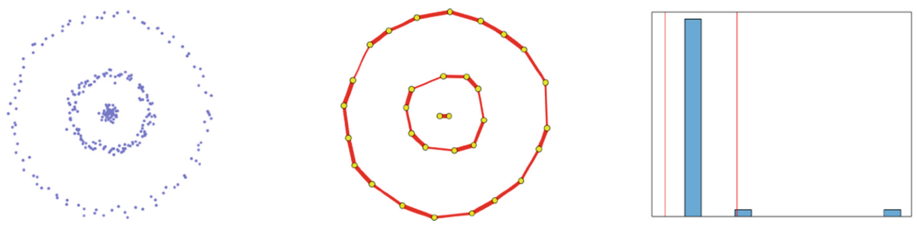

Fig. 2. (left) input data, (middle) $G^* = (V^*, E^*)$ via GNG, (right) histogram for eigenvectors scores, suggesting the top two eigenvectors are sufficient for clustering (best viewed in color).

$$DBI_k(X) + \sum_{i=1}^{k} \mu_i, \qquad 2 \leq k \leq m \tag{15}$$

4 Experiments

The experimental design contains three experiments to test the competing methods. The experiments vary in the input type. The edges weights were assigned using local σ [4] and *CONN* [2]. The local scaling used all the 5 approximation graphs, whereas, *CONN* used only four. *CONN* only needs vector quantization without the edge drawing component. Apart from the second experiment, all methods used an automated selection of k discussed in Sects. 3.3 and 3.4. The experiments were run on a windows 10 machine (3.40 GHz CPU and 8 GB of memory) where methods were coded in MATLAB 2017b.

4.1 Synthetic Data

The competing methods were tested using synthetic data (shown in Fig. 3). The number of representatives m was set by running multiple values using k-means++ algorithm then select the one that represents an elbow point in quantization error curve.

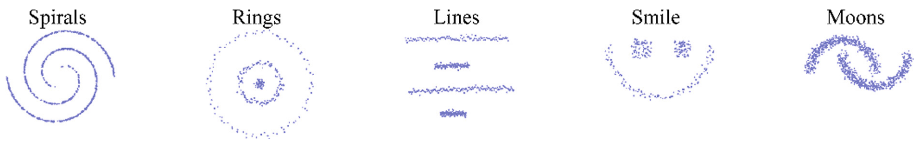

Fig. 3. A collection of synthetic data.

Local σ outcomes in Table 1, suggest that 3NN, 5NN, and SOM are not very useful. However, it could be considerably improved by using edges obtained through NG and GNG. Since these two methods used the competitive Hebbian learning (CHL) [11], they can eliminate edges where the probability of x_i is discontinued. This was extremely helpful for local σ to assign weights on these edges and discard the ones that do not exist, resulting in a better clustering outcome. This demonstrates that the local σ outcome is highly influenced by the quality of the graph. The results were not surprising, it was anticipated by looking at Fig. 1, where 3NN, 5NN, and SOM added unnecessary edges between clusters. On the other hand, NG and GNG provided better graphs that clearly show the separation between clusters.

Moving to the other side of the table where graphs were weighed by *CONN*. It is observable that k-means struggled compared to its peers (i.e. SOM, NG, and GNG). This could be explained that in k-means centroids are moving independently from each other. Therefore, the concept of BMU and second BMU which is required by *CONN*

Table 1. Clustering accuracies for data in Fig. 3. All values are averages of 100.

			local σ					*CONN*			
	n	m	k-means, 3NN	k-means, 5NN	SOM, SOM grid	NG, CHL	GNG, CHL	k-means	SOM	NG	GNG
Spirals	1000	64	0.59 ± 0.1	0.46 ± 0.1	0.37 ± 0.0	0.97 ± 0.1	0.97 ± 0.1	0.85 ± 0.2	0.88 ± 0.1	0.96 ± 0.1	**0.97 ± 0.1**
Rings	299	32	0.89 ± 0.1	0.94 ± 0.1	0.67 ± 0.0	0.92 ± 0.1	**0.97 ± 0.1**	0.88 ± 0.2	0.97 ± 0.1	0.82 ± 0.2	0.95 ± 0.1
Lines	512	64	**0.99 ± 0.0**	0.99 ± 0.1	0.76 ± 0.2	0.98 ± 0.1	**0.99 ± 0.0**	0.86 ± 0.2	0.93 ± 0.2	0.86 ± 0.2	0.87 ± 0.2
Smile	266	32	0.96 ± 0.1	0.99 ± 0.1	0.91 ± 0.1	**0.99 ± 0.0**	0.97 ± 0.1	0.84 ± 0.2	0.95 ± 0.1	0.90 ± 0.1	0.88 ± 0.1
Moons	1000	64	0.89 ± 0.2	0.89 ± 0.2	0.67 ± 0.2	0.95 ± 0.1	0.76 ± 0.2	0.81 ± 0.2	**0.96 ± 0.1**	0.91 ± 0.1	0.89 ± 0.2

was not fully implemented. For other methods, graphs obtained through SOM yields the best performance. While, NG and GNG performed in a similar manner with an advantage for GNG. This could be explained that SOM provided better neurons density than NG and GNG. Neurons density is determined by their adaptation to the input data using Eqs. (6), (7), and (9) for SOM, NG, and GNG respectively.

The outcome of this experiment emphasizes on the efficiency of the local scaling similarity measure if it was passed the appropriate graph. In addition to its small memory footprint, local σ produced the best accuracy in 3 datasets out of 5. Another advantage for local σ over *CONN* is that local σ weighs edges regardless of the density around neurons. This is clearly demonstrated by the performance drop in NG and GNG when they are weighted by *CONN*. On the other hand, local σ could perform poorly when the graph contains unnecessary edges as we saw in 3NN, 5NN, and SOM graphs.

4.2 UCI Datasets

In this experiment, we used datasets retrieved from UCI repository. The automatic estimation of σ discussed in Sects. 3.3 and 3.4 was disabled to reduce the volatility and provide better comparison. For graphs weighted by local σ in Table 2, SOM was the best performer, NG and GNG being closely comparable except for BC-Wisconsin dataset where SOM was considerably higher. On the other hand, graphs weighted by *CONN* did not produce exceptional performance especially in Wine dataset.

Table 2. Clustering accuracies on UCI datasets for 100 runs.

			local σ					*CONN*			
	n	m	k-means, 3NN	k-means, 5NN	SOM, SOM grid	NG, CHL	GNG, CHL	k-means	SOM	NG	GNG
Iris	150	16	0.83 ± 0.1	0.85 ± 0.1	0.78 ± 0.1	0.86 ± 0.1	0.85 ± 0.1	0.86 ± 0.1	**0.88 ± 0.1**	**0.88 ± 0.1**	0.87 ± 0.1
Wine	178	16	0.82 ± 0.1	0.82 ± 0.1	**0.83 ± 0.1**	0.81 ± 0.1	**0.83 ± 0.1**	0.75 ± 0.1	0.72 ± 0.1	0.76 ± 0.1	0.75 ± 0.1
BC-Wisconsin	699	16	0.77 ± 0.1	0.79 ± 0.1	**0.97 ± 0.0**	0.84 ± 0.1	0.83 ± 0.1	0.96 ± 0.0	0.96 ± 0.0	0.95 ± 0.0	0.95 ± 0.0
Segmentation	2100	32	0.48 ± 0.1	0.49 ± 0.1	0.57 ± 0.1	0.55 ± 0.0	0.56 ± 0.0	**0.58 ± 0.1**	0.54 ± 0.1	0.56 ± 0.1	0.56 ± 0.1
Pen Digits	10992	64	0.70 ± 0.1	0.66 ± 0.1	0.73 ± 0.1	0.66 ± 0.0	0.65 ± 0.0	0.73 ± 0.1	**0.75 ± 0.1**	0.68 ± 0.1	0.67 ± 0.1

The memory footprint is where local σ and *CONN* are split apart. In Fig. 4, it is clear that the memory footprint of *CONN* is exponentially increasing with m. This was not the case with local σ where it maintained a linear increase with m. With local σ performing similar to *CONN* in terms of clustering accuracy, it represents a memory efficient option if it was coupled with a topology preserved graph (Fig. 5).

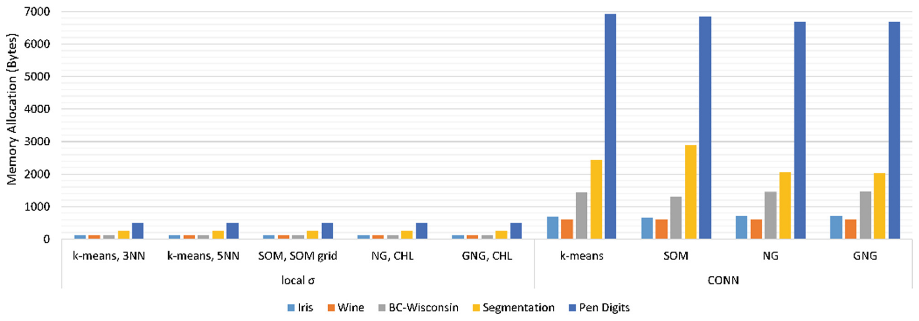

Fig. 4. Similarity measures average memory consumption (best viewed in color).

4.3 Berkeley Segmentation Dataset (BSDS500)

Berkeley Segmentation Dataset (BSDS500) contains 500 images. It uses 3 clustering quality metrics: segmentation covering (covering), rand index (RI), and variation of information (VI). It important to mention that the achieved segmentation results are lower than the numbers reported in the original study [13]. This mainly due to the absence of the edge detection component, which is beyond the scope of this work.

From local σ part in Table 3, one could observe that 3NN, 5NN, and SOM graphs, fluctuated over the clustering metrics. NG and GNG maintained a consistent performance over all measures compared to *CONN* graphs. In terms of covering, they deviated by 0.03 and 0.04 respectively from the best performer. For RI, they were off by 0.06 and 0.08 from the best performer. Finally, for VI, their performance was in line with *CONN* graphs. For *CONN* graphs (on the right side in Table 3), the best performer in terms of covering was NG and the others were not far away from its score. In terms of RI, all methods achieved an equal score. This emphasizes that *CONN* is highly influenced by the edges it draws rather than the vector quantization method used.

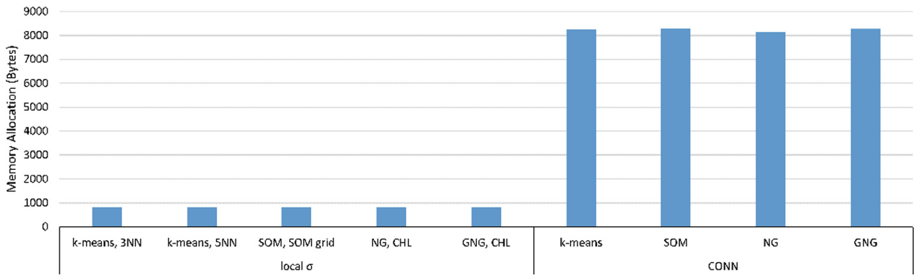

Fig. 5. Average memory needed to compute the similarity measure for methods in Table 3.

Table 3. Image segmentation evaluation using BSDS500

	local σ					*CONN*			
Evaluation metric	k-means, 3NN	k-means, 5NN	SOM, SOM grid	NG, CHL	GNG, CHL	k-means	SOM	NG	GNG
Segmentation covering	0.33	0.36	0.34	0.36	0.35	0.37	0.36	**0.39**	0.38
Rand index	0.59	0.58	0.64	0.62	0.6	**0.68**	**0.68**	**0.68**	**0.68**
Variation of information	3.06	**2.75**	3.03	2.91	2.94	2.91	2.97	2.81	2.83

5 Conclusions

The computational demands for spectral clustering stimulated the research on approximate spectral clustering (ASC). In ASC, learning methods have been effectively used to produce an approximated graph. Nevertheless, prior ASC efforts have either density dependent edges or large memory footprint. In this study we employed topology preserving methods to produce an approximated graph with a preserved topology. The edges were weighed via a locally scaled similarity measure (local σ) that is independent from the density around them. The experiments reveal that local σ coupled with topology preserving graphs could match the performance of recent ASC methods with less memory footprint.

Approximation methods for spectral clustering are known for long preprocessing time to obtain the approximated graph. For future work, we attempt to minimize the preprocessing computations required to produce the approximated graph.

References

1. Yan, D., Huang, L., Jordan, M.I.: Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 907–916 (2009)
2. Tasdemir, K.: Vector quantization based approximate spectral clustering of large datasets. Pattern Recogn. **45**, 3034–3044 (2012)
3. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems (2002)
4. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Proceedings of the 17th International Conference on Neural Information Processing Systems, pp. 1601–1608 (2004)
5. Martinetz, T., Schulten, K.: Topology representing networks. Neural Netw. **7**, 507–522 (1994)
6. Kohonen, T.: The self-organizing map. Proc. IEEE **78**, 1464–1480 (1990)
7. Martinetz, T., Schulten, K.: A “Neural-Gas” Network Learns Topologies. University of Illinois at Urbana-Champaign, Champaign (1991)
8. Fritzke, B.: A growing neural gas network learns topologies. In: Advances in Neural Information Processing Systems, pp. 625–632 (1995)
9. Sugiyama, M.: Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (2007)

10. Zhang, X., Li, J., Yu, H.: Local density adaptive similarity measurement for spectral clustering. Pattern Recogn. Lett. **32**, 352–358 (2011)
11. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: Gielen, S., Kappen, B. (eds.) ICANN 1993, pp. 427–434. Springer, London (1993). https://doi.org/10.1007/978-1-4471-2063-6_104
12. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**, 395–416 (2007)
13. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 898–916 (2011)