Approximate spectral clustering using both reference vectors and topology of the network generated by growing neural gas

Kazuhisa Fujita[1,2]

[1] Komatsu University, Komatsu, Ishikawa, Japan
[2] University of Electro-Communications, Chofu, Tokyo, Japan

ABSTRACT

Spectral clustering (SC) is one of the most popular clustering methods and often outperforms traditional clustering methods. SC uses the eigenvectors of a Laplacian matrix calculated from a similarity matrix of a dataset. SC has serious drawbacks: the significant increases in the time complexity derived from the computation of eigenvectors and the memory space complexity to store the similarity matrix. To address the issues, I develop a new approximate spectral clustering using the network generated by growing neural gas (GNG), called ASC with GNG in this study. ASC with GNG uses not only reference vectors for vector quantization but also the topology of the network for extraction of the topological relationship between data points in a dataset. ASC with GNG calculates the similarity matrix from both the reference vectors and the topology of the network generated by GNG. Using the network generated from a dataset by GNG, ASC with GNG achieves to reduce the computational and space complexities and improve clustering quality. In this study, I demonstrate that ASC with GNG effectively reduces the computational time. Moreover, this study shows that ASC with GNG provides equal to or better clustering performance than SC.

INTRODUCTION

A clustering method is a workhorse and becomes more important for data analysis, data mining, image segmentation, and pattern recognition. The most famous clustering method is k-means, but it cannot accurately partition a nonlinearly separable dataset. Spectral clustering (SC) is one of the efficient clustering methods for a nonlinearly separable dataset (*Filippone et al., 2008*) and can extract even complex structures such as half-moons data (*Bojchevski, Matkovic & Günnemann, 2017*). SC often outperforms traditional popular clustering methods such as k-means (*von Luxburg, 2007*). However, SC has significant drawbacks: the considerable increases in computational complexity and space complexity with the number of data points. These drawbacks make it difficult to use SC for a large dataset. Nowadays, the drawbacks are becoming more crucial as datasets become more massive, more varied, and more multidimensional.

SC treats a dataset as a graph (network) consisting of nodes and weighted edges. The nodes and the edges respectively represent the data points in the dataset and the

Corresponding author
Kazuhisa Fujita,
kazu@spikingneuron.net

connections between the data points. The weights of the edges are the similarities between data points. In SC, eigenvectors are calculated from the Laplacian matrix derived from the similarity matrix of the network. The rows of the matrix that consist of the eigenvectors are clustered using a traditional clustering method such as k-means. The drawbacks of SC for a large dataset are the time complexity to compute the eigenvectors and the space complexity to store the similarity matrix.

Researchers have tackled the high computational cost of SC. There are four approaches to improve the computational cost. The first approach is parallel computing to reduce computational time (*Song et al., 2008*; *Chen et al., 2011*; *Jin et al., 2013*). The second approach is data size reduction by random sampling. For example, *Sakai & Imiya (2009)* make a similarity matrix small by random sampling its columns to reduce the computational cost of eigendecomposition. The third approach is to use a low-rank matrix that approximates the similarity matrix of the original dataset to avoid calculating the whole similarity matrix (*Fowlkes et al., 2004*; *Li et al., 2011*). The last approach is to reduce a data size using a vector quantization method such as k-means (*Yan, Huang & Jordan, 2009*), self-organizing map (SOM) (*Duan, Guan & Liu, 2012*) and neural gas (NG) (*Moazzen & Taşdemir, 2016*). This method is called approximate spectral clustering or two-level approach. In approximate spectral clustering (ASC), data points are replaced with fewer reference vectors. We can decrease the computational cost of SC by reducing data size using a quantization method.

This study focuses on ASC using SOM and its alternatives such as NG and growing neural gas (GNG). SOM and its alternatives are brain-inspired artificial neural networks that represent data points in the network parameters. The network consists of units that have weights regarded as reference vectors and edges connecting pairs of the units. ASC uses fewer reference vectors instead of data points to reduce the size of the input to SC. Furthermore, the reference vectors are regarded as local averages of data points, thus, less sensitive to noise than the original data (*Vesanto & Alhoniemi, 2000*). This may improve the clustering performance of SC. The performance of SC highly depends on the quality of the constructed similarity matrix from input data points (*Chang & Yeung, 2008*; *Li et al., 2018*; *Park & Zhao, 2018*; *Zhang & You, 2011*). In other words, to construct a more robust similarity matrix is to improve the clustering performance of SC (*Lu, Yan & Lin, 2016*). Because of the dependence on the quality of the similarity matrix, SC is highly sensitive to noisy input data (*Bojchevski, Matkovic & Günnemann, 2017*). Thus, the lower sensitivity of reference vectors to noise may improve the clustering quality of SC. In many studies, reference vectors of a network generated by SOM are used to reduce input size, but the topology of the network is not exploited.

In this study, I develop a new ASC using a similarity matrix calculated from both the reference vectors and the topology of the network generated by GNG, called ASC with GNG. The key point is to regard the network generated by GNG as a similarity graph and calculate the similarity matrix from not only the reference vectors but also the topology of the network that reflects the topology of input. I employ GNG to generate a network because the network generated by GNG can represent important topological relationships in a given dataset (*Fritzke, 1994*) and the better similarity matrix will lead to better

clustering performance in the ASC. The quantization by GNG reduces the computational complexity and the space complexity of SC. Furthermore, the effective extraction of the topology by GNG may improve clustering performance. This paper investigates the computation time and the clustering performance of ASC with GNG. Moreover, I compare ASC with GNG with ASC using a similarity matrix calculated from quantization results generated by neural gas, Kohonen's SOM, and k-means instead of GNG.

RELATED WORKS

The most well-known clustering method is k-means (*MacQueen, 1967*), which is one of the top 10 most common algorithms used in data mining (*Wu et al., 2007*). k-means is highly popular because it is simple to implement and yet effective in performance (*Haykin, 2009*). It groups data points so that the sum of Euclidean distances between the data points and their centroids is small. It can accurately partition only linearly separable data.

Expectation–maximization algorithm (EM algorithm) is a popular tool for simplifying difficult maximum likelihood problems (*Hastie, Tibshirani & Friedman, 2009*). EM algorithm can estimate the parameters of a statistical model and is also used for clustering. When we use EM algorithm for clustering, we often assume that data points are generated from a mixture of Gaussian distributions. However, EM algorithm cannot precisely partition data points not generated from the assumed model. k-means is described by slightly extending the mathematics of the EM algorithm to this hard threshold case (*Bottou & Bengio, 1994*) and assumes a mixture of isotropic Gaussian distributions with the same variances.

A major drawback to k-means is that it cannot partition nonlinearly separable clusters in input space (*Dhillon, Guan & Kulis, 2004*). There are two approaches for achieving nonlinear separations using k-means. One is kernel k-means (*Girolami, 2002*) that partitions the data points in a higher-dimensional feature space after the data points are mapped to the feature space using the nonlinear function (*Dhillon, Guan & Kulis, 2004*). All the computation of kernel k-means can be done by the kernel function using the kernel trick (*Ning & Hongyi, 2016*). However, kernel k-means cannot accurately partition data points if the kernel function is not suitable for the data points. The other is k-means using other distances such as spherical k-means (sk-means) (*Dhillon & Modha, 2001*; *Banerjee et al., 2003*, *2005*) and cylindrical k-means (cyk-means) (*Fujita, 2017*). Sk-means uses cosine distance because of assuming the data points generated from a mixture of von Mises–Fisher distributions. Cyk-means uses the distance that combines cosine distance and Euclidean distance because of assuming the data points generated from a mixture of joint distributions of von Mises distribution and Gaussian distribution. However, k-means using other distances cannot accurately partition data points if the distance is not suitable for the data points.

Affinity propagation (*Frey & Dueck, 2007*) is a clustering method using a similarity matrix as in SC and is derived as an instance of the max-sum algorithm. Affinity propagation simultaneously considers all data points as potential exemplars and recursively transmits real-value messages along edges of the network until a good set of exemplars is generated (*Frey & Dueck, 2007*). Its advantage is that there is no need to

pre-specify the number of clusters and no assumption of a mixture of distributions. Its time complexity is $O(N^2T)$ (*Fujiwara, Irie & Kitahara, 2011*; *Khan, Amjad & Kleinsteuber, 2018*), where N and T are the number of data points and the number of iterations, respectively. Space complexity is $O(N^2)$ because it stores a similarity matrix. For a large dataset, these complexities are not ignored.

Spectral clustering (SC) is a popular modern clustering method based on eigendecomposition of a Laplacian matrix calculated from a similarity matrix of a dataset (*Taşdemir, Yalçin & Yildirim, 2015*). SC does not assume a statistical distribution and partitions a dataset using only a similarity matrix. SC displays high performance for clustering nonlinear separable data (*Chin, Mirzal & Haron, 2015*) and has been applied to various fields such as image segmentation (*Eichel et al., 2013*), co-segmentation of 3D shapes (*Luo et al., 2013*), video summarization (*Cirne & Pedrini, 2013*), identification of cancer types (*Chin, Mirzal & Haron, 2015*; *Shi & Xu, 2017*), document retrieval (*Szymański & Dziubich, 2017*). In SC, a dataset is converted to a Laplacian matrix calculated from the similarity matrix of the dataset. We obtain a clustering result by grouping the rows of the matrix that consists of the eigenvectors of the Laplacian matrix. SC often outperforms a traditional clustering method, but it requires enormous computational cost and large memory space for a large dataset. Especially, its use is limited since it is often infeasible due to the computational complexity of $O(N^3)$ (*Izquierdo-Verdiguier et al., 2015*; *Taşdemir, 2012*; *Taşdemir, Yalçin & Yildirim, 2015*; *Wang et al., 2013*), where N is the number of data points. The huge computational complexity of SC is mainly derived from the eigendecomposition and constitutes the real bottleneck of SC for a large dataset (*Izquierdo-Verdiguier et al., 2015*). The required memory space increases with $O(N^2)$ (*Mall, Langone & Suykens, 2013*) because the similarity matrix is an $N \times N$ matrix.

Clustering a large dataset often becomes more challenging due to increasing computational cost with the size of a dataset. One approach to reducing the computational cost of clustering is two-level approach that partitions the quantization result of a dataset (*Vesanto & Alhoniemi, 2000*). In two-level approach, first, data points are converted to fewer reference vectors by a quantization method such as SOM and k-means (abstraction level 1). Then the reference vectors are combined to form the actual clusters (abstraction level 2). Each data point belongs to the same cluster as its nearest reference vector. The two-level approach has the advantage of dealing with fewer reference vectors instead of the data points as a whole, therefore reducing the computational cost (*Brito da Silva & Ferreira Costa, 2014*). This approach does not limit a quantization method in abstraction level 1 and a clustering method in abstraction level 2. There are many pairs of quantization methods and clustering methods, for example, SOM and hierarchical clustering (*Vesanto & Alhoniemi, 2000*; *Taşdemir, Milenov & Tapsall, 2011*), GNG and hierarchical clustering (*Mitsyn & Ososkov, 2011*), and SOM and normalized cut (*Yu et al., 2012*). Especially, the approach using SC in abstraction level 2 is called Approximate Spectral Clustering (ASC) (*Taşdemir, Yalçin & Yildirim, 2015*).

In two-level approach, there is the concern of underperformance of clustering by the quantization because it uses fewer reference vectors instead of all the data points in a

dataset. However, a quantized dataset would be sufficient in many cases (*Bartkowiak et al., 2005*). Furthermore, two-level approach using SOM also has the benefit of noise reduction (*Vesanto & Alhoniemi, 2000*).

APPROXIMATE SPECTRAL CLUSTERING WITH GROWING NEURAL GAS

This paper proposes approximate spectral clustering with growing neural gas (ASC with GNG). ASC with GNG partitions a dataset using a similarity matrix calculated from both reference vectors and the topology of the network generated by GNG. ASC with GNG consists of the two processes, which are abstraction level 1 and abstraction level 2, based on two-level approach (*Taşdemir, Yalçin & Yildirim, 2015*; *Vesanto & Alhoniemi, 2000*). In abstraction level 1, a dataset is converted into the network by GNG. The similarity matrix is calculated from the network, considering the reference vectors and the topology of the network. In abstraction level 2, the reference vectors are merged by SC using the similarity matrix. The data points in the dataset are assigned to the clusters to which the reference vectors nearest to the data points belong.

ASC with GNG can partition a nonlinearly separable dataset. After abstraction level 2, the data points are assigned to the nearest reference vectors based on Euclidean distance. Thus, ASC with GNG divides the space into regions consisting of Voronoi polygons around reference vectors. In ASC with GNG, SC merges the reference vectors to k clusters. Simultaneously, the Voronoi polygons corresponding to the reference vectors are also merged to k regions. ASC with GNG provides the complex decision regions consisting of Voronoi polygons and achieves nonlinear separation.

At first, ASC with GNG appears to be the same as other ASCs because many ASCs using a quantization method such as k-means, SOM and its alternatives already exist. For example, *Duan, Guan & Liu (2012)* have proposed ASC using SOM. In their method, data points are quantized by SOM, and SC partitions the reference vectors. The similarity measure of their method is defined by the Euclidean distance divided by local variance reflecting the distribution of data points around a reference vector. The interesting point of their method is using a local variance. *Moazzen & Taşdemir (2016)* have proposed ASC using NG. Similarly, in their method, the similarity matrix used by SC is calculated from reference vectors. It is interesting that their method uses integrated similarity criteria to improve accuracy. In ASC with GNG as well as the other ASCs, reference vectors are partitioned by SC. However, ASC with GNG uses both reference vectors and topology of the network generated by GNG, but the other ASCs use only reference vectors. Using the topology will play an important role in improving clustering performance because SC can be regarded as a type of partition problem for a network (*Diao, Zhang & Wang, 2015*) and partitions the units of the network into disjoint subsets.

Algorithm

ASC with GNG consists of generating a network by GNG, partitioning reference vectors by SC, and assigning data points into clusters. The algorithm for ASC with GNG is given in Algorithm 1.

Algorithm 1 ASC with GNG.

Input: Data points $X \in R^d$, the number of clusters k

Output: A partition of given data points into k clusters $C_1,\dots,C_k$

1: {Abstraction Level1}

2: Generate the network from the dataset X using GNG.

3: Calculate the similarity matrix $A \in R^{M \times M}$ from the reference vectors and the topology of the network.

4: {Abstraction Level 2}

5: Calculate the normalized Laplacian matrix L_{sym} by Eq. (2).

6: Calculate the k first eigenvectors $\boldsymbol{u}_1,\dots,\boldsymbol{u}_k$ of L_{sym}.

7: Let $U \in R^{M \times k}$ be the matrix containing the vectors $u_1,\dots,u_k$ as columns.

8: For $i = 1,\dots, M$, let $\boldsymbol{y}_i \in R^k$ be the vector corresponding to the i-th row of U.

9: Assign $(\boldsymbol{y}_i)_{i = 1,\dots,M}$ to clusters $C_1,\dots,C_k$ by k-means.

10: {Assign data points to the clusters.}

11: Find the nearest unit to each data point.

12: Each data point is assigned to the cluster to which the nearest unit is assigned.

Let us consider a set of N data points, $X = \{ \boldsymbol{x}_1, \boldsymbol{x}_2,\dots,\boldsymbol{x}_n,\dots,\boldsymbol{x}_N\}$, where $x_n \in R^d$ (Fig. 1A). The data point n is denoted by $\boldsymbol{x}_n$. In this study, each data point is previously normalized by $\boldsymbol{x}_n \leftarrow \boldsymbol{x}_n/\|\boldsymbol{x}_{\text{max}}\|$, where $\boldsymbol{x}_{\text{max}}$ is the data point that has the maximum norm. For an actual application, such normalization will not be required because we can use specific parameters for the application.

GNG generates a network from data points (Fig. 1B). The generated network is not a complete graph, and its topology reflects input space. The unit i in the network has the reference vector $w_i \in R^d$. The algorithm of GNG is denoted in Appendix.

The similarity matrix $A \in R^{M \times M}$ is calculated from the reference vectors and the topology of the generated network. M is the number of units in the network. If the unit i connects to the unit j, the element a_{ij} of A is defined as follows:

$$a_{ij} = \exp\left(-\frac{\|\boldsymbol{w}_i - \boldsymbol{w}_j\|^2}{2\sigma^2} \right), \tag{1}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ are the reference vectors of i and j, respectively. Otherwise, $a_{ij} = 0$. Equation (1) is the Gaussian similarity function that is most widely used to obtain a similarity matrix. The proposed method's new point is to make a similarity matrix from both the reference vectors and the network topology. The similarity matrix is sparse because of using the network topology. However, the similarity matrix has information about significant connections representing input space.

The reference vectors are merged to k clusters by SC as shown in Fig. 1C. The normalized Laplacian matrix $L_{\text{sym}} \in R^{M \times M}$ is calculated from A. Here, we define the diagonal matrix $D \in R^{M \times M}$ to calculate the Laplacian matrix. The element of the diagonal
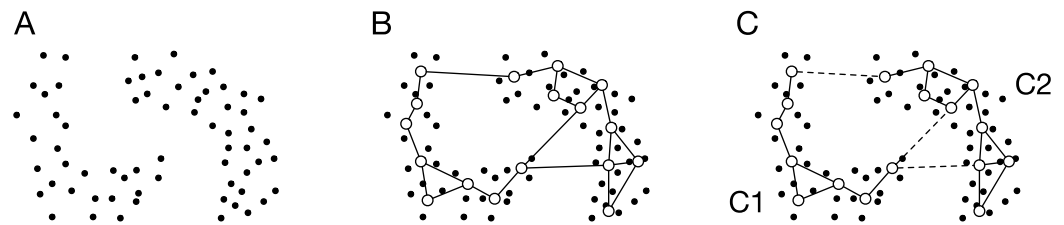
Figure 1 Schematic image of ASC with GNG. (A) Data distribution. The dots denote the data points. (B) Generated network. The open circles and the solid lines denote the units and the edges, respectively. (C) Partition of the units using SC. The broken line denotes the connection cut by SC. SC merges the units into two clusters: C1 and C2. In other words, SC removes the edges connecting the units that belong to different clusters.

matrix is $d_i = \sum_{j=1}^{M} a_{ij}$. The Laplacian matrix is $L = D - A$. We derive the normalized Laplacian matrix from the following equation:

$$L_{\text{sym}} = D^{-1/2}LD^{-1/2}. \tag{2}$$

Here, we calculate the k first eigenvectors $\boldsymbol{u}_1,\dots,\boldsymbol{u}_k$ of L_{sym}. Let $U \in R^{M \times k}$ be the matrix containing the vector $\boldsymbol{u}_1,\dots,\boldsymbol{u}_k$ as columns. For $i = 1,\dots,M$, let $\boldsymbol{y}_i \in R^k$ be the vector corresponding to the i-th row of U. k-means assigns $(\boldsymbol{y}_i)_{i=1,\dots,M}$ to clusters $C_1,\dots,C_k$. Each row vector corresponds with each reference vector one-to-one.

Each data point is assigned to the cluster to which the nearest reference vector belongs. Finally, the data points are assigned to the clusters.

ASC with SOM and its alternatives

In this study, ASC with GNG is compared with ASCs with NG, SOM, k-means, and GNG using no topology to investigate the difference in clustering performances depending on the methods used to generate the network. ASCs with NG and SOM use NG and Kohonen's SOM to generate networks instead of GNG, respectively. The weights of the network generated by NG and SOM are calculated using the Gaussian similarity function. ASC with k-means uses centroids generated by k-means as units in the network. The network is fully connected, and its weights are calculated using the Gaussian similarity function. ASC with GNG using no topology uses only the reference vectors generated by GNG. In ASC with GNG using no topology, the network is fully connected, and its weights are calculated using the Gaussian similarity function. In this study, ASC with GNG is also compared with SC. SC uses a fully-connected network, the Gaussian similarity function, a normalized Laplacian matrix $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$, and k-means.

Complexity

The time complexity of SC is $O(N^3)$ (*Izquierdo-Verdiguier et al., 2015*; *Taşdemir, Yalçin & Yildirim, 2015*; *Wang et al., 2013*), where N is the number of data points. The complexity relies on the eigendecomposition of a Laplacian matrix. The eigendecomposition takes $O(N^3)$ time.

The time complexity of ASCs with GNG is $O(MT + M^3 + NM)$, where M and T are respectively the number of units and the number of iterations. $O(MT)$ is the time

complexity of GNG because $O(M)$ is required to find the best match unit of a presented data point at every iteration. To decide the best match unit, we need to compute the distance between every reference vector and a presented data point and find the minimum distance. Thus, we calculate and compare M distances at every iteration. $O(M^3)$ is the time complexity of SC to partition the units. $O(MN)$ is the time complexity to assign data points to clusters. To assign data points to clusters, we determine the nearest reference vector of every data point. Thus, we calculate M distances for each data point. For $N \gg M$, $O(MN)$ dominates the computational time of ASCs with GNG.

If we use a sort algorithm, such as quicksort, to simultaneously find the best match unit and the second-best match unit every iteration, the time complexity of ASC with GNG is $O(T M \log M + M^3 + NM)$. In this case, GNG sorts the distances between all the reference vectors and a presented data point to finds the best match and the second-best match units at every iteration. The sort algorithm takes $O(M \log M)$.

The time complexity of ASC with NG is $O(TM \log M + M^3 + NM)$. The complexity of NG is $O(TM \log M)$. In NG, the distances between all the reference vectors and a presented data point are calculated at every iteration. The calculation of the distances takes $O(M)$. NG sorts the distances to obtain the neighborhood ranking of the reference vectors at every iteration. The sort of M distances takes $O(M \log M)$ when we use quicksort. NG updates the weights of all the reference vectors at every iteration. NG takes $O(M)$ to update the weighs. Thus, the time complexity of NG is $O(T(M + M \log M + M)) = O(TM \log M)$. For $N \gg M$, $O(MN)$ dominates the computational time of ASC with NG.

The time complexity of ASC with SOM is $O(MT + M^3 + NM)$. $O(MT)$ is the time complexities of SOM because it requires finding the best match unit of a presented data point and updating all the reference vectors at every iteration. For $N \gg M$, $O(MN)$ dominates the computational time of ASCs with SOM.

The time complexities of ASC with k-means is $O(MNT_{kmeans} + M^3 + NM)$, where T_{kmeans} is the number of iterations in k-means. $O(MNT_{kmeans})$ is the time complexities of k-means because the distances between M centroids and N data points are calculated at every iteration. It is difficult to estimate T_{kmeans}, but the convergence of k-means is fast (*Bottou & Bengio, 1994*) and T_{kmeans} will be enough smaller than N. For $N \gg M$, $O(MNT_{kmeans} + NM)$ dominates the computational time of ASCs with k-means. In this case, the computational time approximately linearly increases with N because T_{kmeans} will be enough smaller than N.

The space complexity of SC is $O(N^2)$ (*Mall, Langone & Suykens, 2013*) because the memory space is required to store the $N \times N$ similarity matrix. The space complexity of ASCs with GNG, NG, SOM, and k-means is $O(N + M^2)$. $O(N)$ is the memory space to store the data points. $O(M^2)$ is the memory space to store the similarity matrix of reference vectors. For $N \gg M$, the space complexity of the ASCs is $O(N)$. Thus, the space complexities of the ASCs are much smaller than that of SC.

The time complexities and the space complexities of ASCs and SC are summarized in Table 1.

Table 1 Time complexities and space complexities.

Method	Time complexity	Space complexity
ASC with GNG	$O(MT + M^3 + NM)$	$O(N + M^2)$
ASC with NG	$O(TM \log M + M^3 + NM)$	$O(N + M^2)$
ASC	$O(MT + M^3 + NM)$	$O(N + M^2)$
ASC with k-means	$O(MNT_{kmeans} + M^3 + NM)$	$O(N + M^2)$
SC	$O(N^3)$	$O(N^2)$

Parameters

All the parameters of the methods, except for T, M_{max}, M, and l, are found using grid search to maximize the mean of the purity scores for the datasets used in "Clustering Quality". The parameters of ASC with GNG are $T = 10^5$, $\lambda = 250$, $\varepsilon_1 = 0.1$, $\varepsilon_n = 0.01$, $a_{max} = 75$, $\alpha = 0.25$, $\beta = 0.99$, $\sigma = 0.25$. The parameters of ASC with NG are $T = 10^5$, $\lambda_i = 1.0$, $\lambda_f = 0.01$, $\varepsilon_i = 0.5$, $\varepsilon_f = 0.005$, $a_{maxi} = 100$, $a_{maxf} = 300$, and $\sigma = 0.25$. The Parameters of ASC with SOM are $T = 10^5$, $\gamma_0 = 0.05$, $\sigma_0 = 1.0$, and $\sigma = 0.5$. The Parameter of ASC with k-means is $\sigma = 0.1$. The parameters of ASC with GNG no edge are $T = 10^5$, $\lambda = 350$, $\varepsilon_1 = 0.05$, $\varepsilon_n = 0.01$, $a_{max} = 100$, $\alpha = 0.5$, $\beta = 0.999$, $\sigma = 0.5$. The Parameter of ASC with k-means and SC is $\sigma = 0.1$. The meanings of the parameters are shown in "Algorithm" and Appendix.

RESULTS

This section describes computational time and clustering performances of ASCs with GNG.

The programs used in the experiments were implemented using Python and its libraries. The libraries are numpy for linear algebra computation, networkx for dealing a network, and scikit-learn for loading datasets and using k-means.

Computational time

This subsection describes the computational time of ASCs with GNG, NG, SOM, and k-means and SC. For the measurement of the computational time, the dataset has five clusters and consists of 3-dimensional data points. This dataset is generated by datasets. make_blobs, which is the function of scikit-learn. The dataset is called Blobs in this study. In this experiment, I use the computer with two Xeon E5-2687W v4 CPUs and 64 GB of RAM and cluster the dataset using only one thread to not process in parallel.

Figure 2A shows the relationship between the computational time and the number of data points for 10^2 units. The computational time of ASC with k-means is shortest under about 5×10^5 data points. The computational time of ASC with k-means linearly increases from about 10^4 data points. ASCs with GNG, NG, and SOM show better computational performance than ASC with k-means beyond about 10^6 data points. The computational time of ASCs with GNG, NG, and SOM does not significantly change under about 10^5 data points. However, the computational time of ASCs with GNG, NG, and SOM linearly increases with the number of data points from about 10^6 data points.
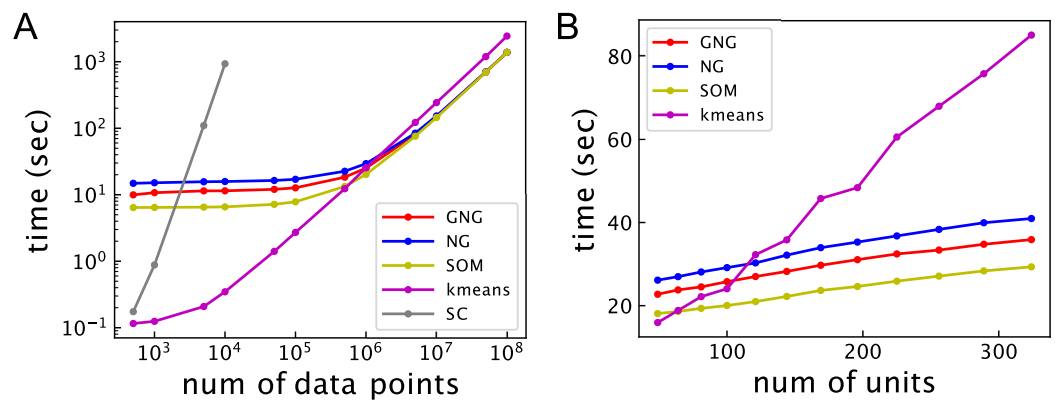
Figure 2 Computational time of ASCs for clustering the synthetic dataset. (A) Relationship between computational time and the number of data points. The number of units of all ASCs is 10^2. (B) Relationship between computational time and the number of units. The number of the data points in the dataset is 10^6. GNG, NG, SOM, k-means, and SC in these figures represent the computational time of ASCs with GNG, NG, SOM, and k-means and SC, respectively. The computational time is the mean of 10 runs with random initial values.

Under 10^5 data points, the computational time of ASCs with GNG, NG, and SOM will mainly depend on the computational cost to generate a network because $O(MT)$ and $O(TM \log M)$ are more than $O(M^3)$ and $O(MN)$. On the other hand, the linear increase of the computational time of ASCs with GNG, NG, and SOM will be dominantly derived from the computational cost to assign data points to units. The computational time of ASC with SOM is shorter than that of ASCs with GNG and NG under about 10^6 data points because SOM does not have the process to change the network topology. From about 10^7 data points, the computational time of ASC with SOM is not different from ASCs with GNG and NG because the computational cost to assign data points to units is dominant. Under 10^6 data points, the computational time of ASC with GNG is shorter than ASC with NG. This difference occurs because the mean of growing M of ASC with GNG during learning is less than M of ASC with NG. The computational time of SC is not shown for more than 10^4 data points because the computational time of spectral clustering is too long compared to the other methods. These results show that the ASCs outperform SC in terms of computational time for a large dataset. However, the results also show that ASC using topology is not effective for a small dataset.

Figure 2B shows the relationship between the computational time and the number of units for 10^6 data points. In ASC with GNG, the number of units means the maximum of the number $M_{\max}$. The computational time of all ASCs linearly increases with the number of units. The computational time of the ASCs with GNG, NG, and SOM increases more slowly than k-means because the complexity of k-means depends on not only the number of units but also the number of data points.

Clustering quality

To investigate the clustering performances of the ASC with GNG, NG, and SOM, three synthetic datasets and six real-world datasets are used. The synthetic datasets are Blobs, Circles, and Moons generated by datasets.make_blobs, datasets.make_circles, and datasets.

Table 2 Datasets.

Dataset	k	n	d
Blobs	3	1,000	2
Circles	2	1,000	2
Moons	2	1,000	2
Iris	3	150	4
Wine	3	178	13
Spam	2	4,601	54
CNAE-9	9	1,080	856
Digits	10	1,797	8×8
MNIST	10	70,000	28×28

Note:
  k, n, and d indicate the number of classes, data points and attributions, respectively.

make_moons that are functions of scikit-learn, respectively. Blobs are generated from three isotropic Gaussian distributions. The standard deviation and the means of each Gaussian are default values of the generating function. Blobs can be partition by a linear separation method such as k-means. Circles consists of two concentric circles. The noise and the scale parameters of the function generating Circles are set at 0.05 and 0.5, respectively. Moons includes two-moons shape distributed data points. The noise parameter of the function generating Moons is 0.05. Circles and Moons are typical synthetic datasets that can not be partitioned by a linear separation method. The real-world data are Iris, Wine, Spam, CNAE-9, Digits, and MNIST (*Lecun et al., 1998*). Iris, Wine, Spam, CNAE-9, and Digits are datasets found in the UCI Machine Learning Repository. In this study, Iris, Wine, Digits, and MNIST are obtained using scikit-learn. Iris and Wine are datasets frequently used to evaluate the performance of a clustering method. Spam and CNAE-9 are word datasets. In this study, three attributions of Spam: capital_run_length_average, capital_run_length_longest, and capital_run_length_total, are not used. Digits and MNIST are handwritten digits datasets. Table 2 shows the numbers of classes, data points, and attributions of the datasets.

To evaluate the clustering methods, we use the purity score. Purity is given by $\text{Purity} = 1/N \sum_{i=1}^{k} \max_j n_i^j$, where N is the number of data points in a dataset, k is the number of clusters, and n_i^j is the number of data points that belong to the class j in the cluster i. When the purity score is 1, all data points belong to true clusters.

Table 3 shows the purities of ASCs with GNG, NG, SOM, k-means, and GNG using no topology and SC. ASC with GNG shows the best accurate clustering results for five datasets: Circles, Moons, Spam, CNAE-9, and Digits. For Blobs, ASC with GNG displays the second-best performance, but the difference of purity between ASC with GNG, NG, and GNG using no topology and SC is small. For MNIST, ASC with GNG shows the second-best performance. ASC with NG also shows relatively high purities for Blobs, Circles, Moons, and Digits. However, for Iris and Wine dataset, the clustering performances of ASCs with GNG and NG are worse than the others. The lower performance of ASCs with GNG and NG may be caused by too many units for the number

Table 3 Performances of clustering (purity).

Dataset	GNG	NG	SOM	k-means	GNG using no topology	SC
Blobs	0.9744	0.9632	0.4110	0.9690	**0.9813**	0.9671
Circles	**1.0000**	**1.0000**	0.5403	0.9997	**1.0000**	**1.0000**
Moons	**0.9992**	0.9884	0.5810	0.9328	0.9985	0.9933
Iris	0.5840	0.5648	0.5020	0.8473	0.8427	**0.8533**
Wine	0.4650	0.4649	0.4379	0.6656	**0.6827**	0.6742
Spam	**0.7676**	0.6063	0.6082	0.6095	0.7464	0.6070
CNAE-9	**0.6711**	0.5920	0.2913	0.4887	0.5706	0.1871
Digits	**0.8572**	0.8129	0.3356	0.7641	0.8025	0.6023
MNIST	0.6100	0.5801	0.2784	**0.6754**	0.5888	nan

Note:
GNG, NG, SOM, k-means, GNG using no topology, and SC mean ACSs with GNG, NG, SOM, k-means, and GNG using no topology, and spectral clustering, respectively. The purities are the mean of 100 runs with random initial values. The best purities are bold.

of data points. Perhaps, there will be the optimal number of units to make a better similarity matrix. The performance of ASC with SOM is worse than the other methods for the datasets apart from Spam. The bad performance of ASC with SOM may be caused by the feature of SOM that is the tendency to have null units often. This feature of SOM is unsuitable for ASC using topology. ASC with k-means shows relatively high performance for Blobs, Circles, Moons, Iris, Wine, and MNIST. For MNIST, the performance of ASC with k-means is best. ASC with GNG using no topology shows the best performance for Blobs, Circles, and Wine. For Moons and Spam, ASC with GNG using no topology displays the second-best performance. For Blobs, Circles, Moons, Wine, Spam, CNAE-9, and Digits, the performance of ASC with GNG using no topology is better than that of ASC with k-means. This result suggests that GNG can quantize dataset better than k-means in many cases. For MNIST, SC cannot perform clustering because overflow occurs. These results suggest that the network topology effectively improves the performance of the clustering, and GNG will generate the same or better quantization result than k-means.

Figure 3 shows the relationship between purity and the number of units for Circles, Iris, and MNIST to investigate the dependence of clustering performance on the number of units. For Circles, the clustering performances of ASCs with GNG and NG vary in convex upward. This result suggests that the performances become worse for the larger or smaller number of units than the optimal number of units and become better for the around optimal number of units. For Iris, the performances of ASCs with GNG and NG are high under about 50 and 36 units, respectively. From about 50 and 36 units, these performances go down with the number of units. This result suggests that the number of units used in Table 3 is too large to obtain the optimal performances for Iris and Wine. For MNIST, the performances of ASCs with GNG and NG improve with the number of units. For all cases, the performances of ASCs with k-means and GNG using no topology more weakly depend on the number of units. This result suggests that ASC not using network
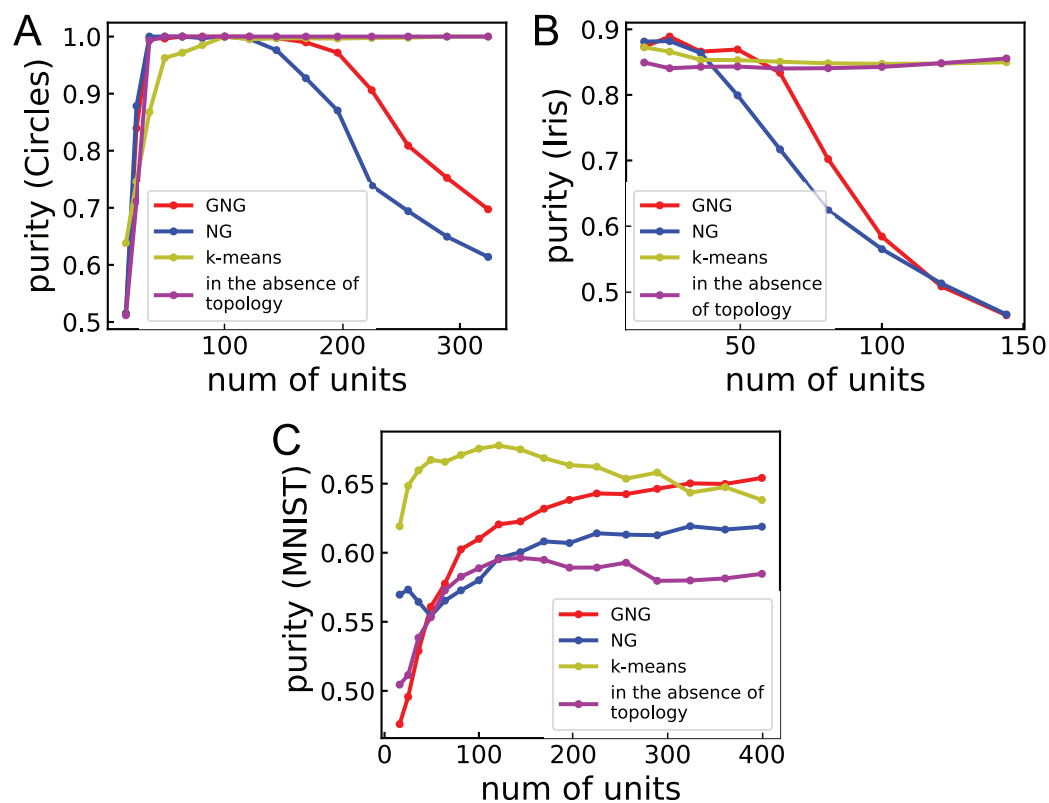
Figure 3 The relationship between purity and the number of units. (A)–(C) Show the performance of clustering Circles, Iris, and MNIST, respectively. Purities are the mean of 100 runs with random initial values. The line labeled "in the absence of topology" denotes the clustering performance of ASC with GNG using no topology.

topology displays peak performance less than ASCs using network topology in many cases but its performance does not strongly depend on the number of units.

CONCLUSION

This study proposes approximate spectral clustering using the network generated by growing neural gas, called ASC with GNG. ASC with GNG partitions a dataset using a Laplacian matrix calculated from not only the reference vectors but also the topology of the network generated by GNG. ASC with GNG displays better computational performance than SC for a large dataset. Furthermore, the clustering quality of ASC with GNG is equal or better than SC in many cases. The results of this study suggest that ASC with GNG improves not only computational but also clustering performances. Therefore, ASC with GNG can be a successful method for a large dataset.

Why does ASC with GNG display better clustering performance? The clustering results of SC depend on the quality of the constructed network from which a Laplacian matrix is calculated (*Li et al., 2018*). Namely, we need to improve the way to construct a similarity matrix to obtain better clustering performance for SC (*Lu, Yan & Lin, 2016*; *Park & Zhao, 2018*). The network generated by GNG represents the important topological

relationships in a given dataset (*Fritzke, 1994*). Furthermore, in ASC with GNG, a similarity matrix calculated using the network topology is sparse because the elements of a similarity matrix between not connected units are zero. The sparse similarity matrix may contribute to the improvement of clustering performance. Therefore, the ability of GNG to extract the topology of a dataset will lead to improving clustering performance.

However, when the number of units is not optimal, ASC with GNG will produce more unsatisfactory results than ASC with GNG using no topology, and SC. This problem occurs when the number of units is close to or too smaller than the number of data points. This problem will not frequently occur in the actual application dealing with a large dataset.

ASC using no topology, such as ASCs with k-means and GNG using no topology, also displays high performance, but its peak performance is less than ASC with GNG in many cases. If we are concerned about the dependence of performance on the number of units, then ASC using no topology may also be a good option.

Why can ASC partition a nonlinearly separable dataset? The ability for nonlinear clustering is provided by SC used in abstraction level 2. Simply put, abstraction level 1 reduces the size of a dataset. The assignment after abstraction level 2 does not provide nonlinear separation, as only it creates only Voronoi regions around the reference vectors. SC makes the complex decision region merging the Voronoi regions and achieves nonlinear clustering. Thus, the ability for nonlinear clustering is not derived from the way of abstraction level 1. However, the clustering performance of ASC also depends on the way of abstraction level 1, as mentioned in the results section.

ASC with GNG can be more accelerated by parallel computing. *García-Rodríguez et al. (2011)* have achieved to accelerate GNG using graphics processing unit (GPU). *Vojácek & Dvorský (2013)* have parallelized GNG algorithm using high-performance computing. Thus, the process of making the network using GNG can be accelerated by GPU or high-performance computing. Furthermore, finding the nearest reference vectors of data points can easily be parallelized because each calculation of distance is independent. If the nearest reference vectors are found using parallel computing with p threads, the computational time of the finding is reduced to $1/p$.

SC and ASC have a limitation as well as other clustering methods such as k-means. SC and ASCs can make nonlinear boundaries but can only group neighboring data points based on similarity or distance. In other words, they cannot group some not-neighboring subclusters that represent the same category. For instance, in MNIST, the subclusters representing the same digit are scattered, as shown in Fig. 10 of *Khacef et al. (2019)*. To accurately cluster such a dataset, the scattered subclusters must be merged into one cluster. *Khacef et al. (2019)* and *Khacef, Rodriguez & Miramond (2020)* have accurately clustered MNIST using only six hundred labeled data points. Their proposed method projects input data onto a 2-dimensional feature map using SOM and then labels reference vectors using post-labeled unsupervised learning. Post-labeled unsupervised learning using SOM is efficient if we have some labeled data points in a dataset and require a more accurate clustering result.

APPENDIX

A self-organizing map and its alternatives

Self-organizing map (SOM) and its alternatives, such as NG and GNG, are artificial neural networks using unsupervised learning. They convert data points to fewer weights (representative vectors) of units in a neural network and preserve the input topology as the topology of the neural network. Kohonen's SOM (*Kohonen, 1990*) is a basic and typical SOM algorithm. Kohonen's SOM has the features that the network topology is fixed into a lattice (*Sun, Liu & Harada, 2017*) and the number of units is constant. NG has been proposed by *Martinetz & Schulten (1991)* and can flexibly change the topology of its network. However, in NG, we have to preset the number of units in the network. GNG (*Fritzke, 1994*) achieves to flexibly change both the network topology and the number of units in the network according to an input dataset. GNG can find the topology of an input distribution (*Garca-RodrGuez et al., 2012*). GNG has been widely applied to clustering or topology learning such as extraction of two-dimensional outline of an image (*Angelopoulou, Psarrou & García-Rodríguez, 2011*; *Angelopoulou et al., 2018*), reconstruction of 3D models (*Holdstein & Fischer, 2008*), landmark extraction (*Fatemizadeh, Lucas & Soltanian-Zadeh, 2003*), object tracking (*Frezza-Buet, 2008*), and anomaly detection (*Sun, Liu & Harada, 2017*).

Kohonen's self-organizing map

Kohonen's SOM is one of the neural network algorithms and a competitive learning and unsupervised learning algorithm. Kohonen's SOM can project high-dimensional to a low-dimensional feature map. The function of the projection is used for cluster analysis, visualization, and dimension reduction of a dataset.

The topology of the network of Kohonen's SOM is a two-dimensional $l \times l$ lattice in this study, where $l \times l = M$. The unit i in the network has the reference vector $\boldsymbol{w}_i \in R^d$. The unit i is at $\boldsymbol{p}_i \in R^2$ on lattice, where $\boldsymbol{p}_i = ((\mod(i - 1, l) + 1)/l, \lceil i/l \rceil /l)$, $i = \{1, 2, \ldots, M\}$. $\mod(a, b)$ is the remainder of the division of a by b. A general description of SOM algorithm is as follows:

1. Initialize the reference vectors of the units. All elements of the reference vectors are randomly initialized in the range of [0, 1].

2. Randomly select a data point $\boldsymbol{x}_n$ and find the best match unit c, that is

$$c = \arg \min_i \|\boldsymbol{x}_n - \boldsymbol{w}_i\|. \tag{3}$$

3. Update the reference vectors of all units. The new reference vector of the unit i is defined by

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + h_{ci}(t)(\boldsymbol{x}_n - \boldsymbol{w}_i), \tag{4}$$

where t is the number of iterations, $h_{ci}(t)$ is the neighborhood function. $h_{ci}(t)$ is described by the following equation:

$$h_{ci}(t) = \alpha(\gamma_0, t) \exp\left(-\frac{\mathrm{sqdist}(i, c)}{2\alpha(\sigma_0, t)^2}\right), \tag{5}$$

where $\alpha(z, t)$ is a monotonically decreasing scalar function of t, and sqdist(i, c) is the square of the geometric distance between the unit i and the best match unit c on the lattice. $\alpha(z, t)$ is defined by

$$\alpha(z, t) = z \times \left(1 - \frac{t}{T}\right), \tag{6}$$

sqdist(i, c) is defined by

$$\text{sqdist}(i, c) = \|\boldsymbol{p}_i - \boldsymbol{p}_c\|^2. \tag{7}$$

4. If $t = T$, terminate. Otherwise, go to Step 2.

Neural gas (NG)

NG also generates a network from input data points. NG flexibly changes the topology of the network according to input data points, but the number of the units is static. The network consists of M units and edges connecting pairs of units. The unit i has the reference vector $\boldsymbol{w}_i$. The edges are not weighted and not directed. An edge has a variable called age to decide whether the edge is deleted. Let us consider the set of N data points, $X = \{ \boldsymbol{x}_1, \boldsymbol{x}_2,\ldots,\boldsymbol{x}_n,\ldots,\boldsymbol{x}_N\}$, where $x_n \in R^d$. The algorithm of the neural gas is shown below:

1. Assign initial values to the weight $\boldsymbol{w}_i \in R^d$ and set all C_{ij} to zero. C_{ij} describes the connection between the unit i and the unit j.
2. Select a data point $\boldsymbol{x}_n$ from the dataset at random.
3. Determine the neighborhood-ranking of i, k_i, according to distance between $\boldsymbol{w}_i$ and $\boldsymbol{x}_n$ by the sequence of ranking $(i_0, i_1,\ldots,i_k,\ldots,i_{M-1})$ of units with

$$\|\boldsymbol{x} - \boldsymbol{w}_{i_0}\| < \|\boldsymbol{x} - \boldsymbol{w}_{i_1}\| < \ldots < \|\boldsymbol{x} - \boldsymbol{w}_{i_k}\| < \ldots < \|\boldsymbol{x} - \boldsymbol{w}_{i_{N-1}}\|. \tag{8}$$

4. Perform as adaptation step for the weights according to

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \varepsilon e^{-k_i/\lambda}(\boldsymbol{x}_n - \boldsymbol{w}_i), i = 1, \ldots, N. \tag{9}$$

5. Determine the nearest neighbor unit i_0 and the second nearest neighbor unit i_1. If $C_{i_0 i_1} = 0$, set $C_{i_0 i_1} = 1$ and $l_{i_0 i_1} = 0$. If $C_{i_0 i_1} = 1$, set $l_{i_0 i_1} = 0$. $l_{i_0 i_1}$ describes the age of the edge between the unit i_0 and the unit i_1.

6. Increase the age of all connections of i_0 by setting $l_{i_0 j} = l_{i_0 j} + 1$ for all j with $C_{i_0 j} = 1$.
7. Remove all connections of i_0 that have their age exceeding the lifetime by setting $C_{i_0 j} = 0$ for all j with $C_{i_0 j} = 1$ and $l_{i_0 j} > a_{\max}$.
8. If the number of iterations is not T, go to step 2.

ε, λ, and $a_{\max}$ decay with the number of iterations t. This time dependence has the same form for these parameters and is determined by $g(t) = g_i \, (g_f/g_i)^{t/T}$.

Growing neural gas

Growing neural gas (GNG) can generate a network from a given set of input data points. The network represents important topological relations in the data points using Hebb-like learning rule (*Fritzke, 1994*). A network generated by GNG consists of units and edges that are connections between units. In GNG, not only the number of the units but also the topology of the network can flexibly change according to input data points. The unit i has the reference vector $w_i \in R^d$ and summed error E_i. The edges are not weighted and not directed. An edge has a variable called age to decide whether the edge is deleted.

Let us consider the set of N data points, $X = \{ \boldsymbol{x}_1, \boldsymbol{x}_2,\ldots,\boldsymbol{x}_n,\ldots,\boldsymbol{x}_N \}$, where $\boldsymbol{x}_n \in R^d$. The algorithm of the GNG to make a network from X is given by the following:

1. Start the network with only two units that are connected to each other. The reference vectors of the units set two data points randomly selected from X.

2. Select a data point $\boldsymbol{x}_n$ from the dataset at random.

3. Find the winning unit s_1 of $\boldsymbol{x}_n$ by

$$s_1 = \arg\min_i \|\boldsymbol{x}_n - \boldsymbol{w}_i\|. \tag{10}$$

Simultaneously, find the second nearest unit s_2.

4. Add the squared distance between $\boldsymbol{x}_n$ and $\mathbf{w}_{s_1}$ to the summed error E_{s_1}:

$$E_{s_1} \leftarrow E_{s_1} + \|\boldsymbol{x}_n - \boldsymbol{w}_{s_1}\|. \tag{11}$$

5. Move $\mathbf{w}_{s_1}$ toward $\boldsymbol{x}_n$ by fraction ε_{s_1} of the total distance:

$$\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \varepsilon_{s_1}(\boldsymbol{x}_n - \boldsymbol{w}_{s_1}). \tag{12}$$

Also move the reference vectors of the all direct neighbor units s_n of s_1 toward $\boldsymbol{x}_n$ by the fraction ε_{s_n} of the total distance:

$$\boldsymbol{w}_{s_n} \leftarrow \boldsymbol{w}_{s_n} + \varepsilon_{s_n}(\boldsymbol{x}_n - \boldsymbol{w}_{s_n}). \tag{13}$$

6. If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If s_1 and s_2 are not connected, create the edge connecting between these units.

7. Add one to the ages of all the edges emanating from s_1.

8. Remove the edges with their age larger than $a_{\max}$. If this results in nodes having no emanating edges, remove them as well.

9. Every certain number λ of the input data point generated, insert a new unit as follows:

- Determine the unit q with the maximum summed error E_q.
- Find the node f with the largest error among the neighbors of q.
- Insert a new unit r halfway between q and f as follows:

$$w_r = (w_q + w_f)/2. \tag{14}$$

The number of units has the limit M_s.

- Insert edges between r and q, and r and f. Remove the edge between q and f.
- Decrease the summed errors of q and f by multiplying them with a constant α. Initialize the summed error of r with the new summed error of q.

10. Decrease all summed errors by multiplying them with a constant β

11. If the number of iterations is not T, go to step 2.

Funding

The author received no funding for this work.

Competing Interests

The author declares that they have no competing interests.

Author Contributions

- Kazuhisa Fujita conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

Python scripts are available at GitHub: https://github.com/KazuhisaFujita/asc-using-som. The account and the repositories are KazuhisaFujita and asc-using-som, respectively.

REFERENCES

Angelopoulou A, García-Rodríguez J, Orts-Escolano S, Gupta G, Psarrou A. 2018. Fast 2d/3d object representation with growing neural gas. *Neural Computing and Applications* **29(10)**:903–919 DOI 10.1007/s00521-016-2579-y.

Angelopoulou A, Psarrou A, García-Rodríguez J. 2011. A growing neural gas algorithm with applications in hand modelling and tracking. In: Cabestany J, Rojas I, Joya G, eds. *Advances in Computational Intelligence*. Berlin: Springer, 236–243.

Banerjee A, Dhillon I, Ghosh J, Sra S. 2003. Generative model-based clustering of directional data. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data, KDD '03*. New York: ACM, 19–28.

Banerjee A, Dhillon IS, Ghosh J, Sra S. 2005. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research* **6**:1345–1382.

Bartkowiak A, Zdziarek J, Evelpidou N, Vassilopoulos A. 2005. Choosing representative data items: Kohonen, neural gas or mixture model? In: *Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems*. Boston: Springer, 337–344.

Bojchevski A, Matkovic Y, Günnemann S. 2017. Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings. In: *Proceedings of the 23rd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 737–746.

Bottou L, Bengio Y. 1994. Convergence properties of the k-means algorithms. In: *Proceedings of the 7th International Conference on Neural Information Processing Systems, NIPS'94*. Cambridge: MIT Press, 585–592.

Brito da Silva LE, Ferreira Costa JA. 2014. A density-based clustering of the self-organizing map using graph cut. In: *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. Piscataway: IEEE, 32–40.

Chang H, Yeung D-Y. 2008. Robust path-based spectral clustering. *Pattern Recognition* **41(1)**:191–203 DOI 10.1016/j.patcog.2007.04.010.

Chen W, Song Y, Bai H, Lin C, Chang EY. 2011. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33(3)**:568–586 DOI 10.1109/TPAMI.2010.88.

Chin AJ, Mirzal A, Haron H. 2015. Spectral clustering on gene expression profile to identify cancer types or subtypes. *Jurnal Teknologi* **76**:289–297.

Cirne MVM, Pedrini H. 2013. A video summarization method based on spectral clustering. In: Ruiz-Shulcloper J, Sanniti di Baja G, eds. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Berlin: Springer, 479–486.

Dhillon IS, Guan Y, Kulis B. 2004. Kernel k-means: Spectral clustering and normalized cuts. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, 551–556.

Dhillon IS, Modha DS. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning* **42(1–2)**:143–175.

Diao C, Zhang A-H, Wang B. 2015. Spectral clustering with local projection distance measurement. *Mathematical Problems in Engineering* **2015**:1–13.

Duan Y, Guan T, Liu L. 2012. Self-organizing map based multiscale spectral clustering for image segmentation. In: *2012 International Conference on Computer Science and Electronics Engineering*. **1**:329–333.

Eichel JA, Wong A, Fieguth P, Clausi DA. 2013. Robust spectral clustering using statistical sub-graph affinity model. *PLOS ONE* **8(12)**:e82722 DOI 10.1371/journal.pone.0082722.

Fatemizadeh E, Lucas C, Soltanian-Zadeh H. 2003. Automatic landmark extraction from image data using modified growing neural gas network. *IEEE Transactions on Information Technology in Biomedicine* **7(2)**:77–85.

Filippone M, Camastra F, Masulli F, Rovetta S. 2008. A survey of kernel and spectral methods for clustering. *Pattern Recognition* **41(1)**:176–190 DOI 10.1016/j.patcog.2007.05.018.

Fowlkes C, Belongie S, Chung F, Malik J. 2004. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26(2)**:214–225 DOI 10.1109/TPAMI.2004.1262185.

Frey BJ, Dueck D. 2007. Clustering by passing messages between data points. *Science* **315(5814)**:972–976 DOI 10.1126/science.1136800.

Frezza-Buet H. 2008. Following non-stationary distributions by controlling the vector quantization accuracy of a growing neural gas network. *Neurocomputing* **71(7–9)**:1191–1202 DOI 10.1016/j.neucom.2007.12.024.

Fritzke B. 1994. A growing neural gas network learns topologies. In: *Proceedings of the 7th International Conference on Neural Information Processing Systems, NIPS'94*. Cambridge: MIT Press, 625–632.

Fujita K. 2017. A clustering method for data in cylindrical coordinates. *Mathematical Problems in Engineering* **2017(3)**:1–11 DOI 10.1155/2017/3696850.

Fujiwara Y, Irie G, Kitahara T. 2011. Fast algorithm for affinity propagation. In: *IJCAI'11*. AAAI Press, 2238–2243.

Garca-RodrGuez J, Angelopoulou A, Garca-Chamizo JM, Psarrou A, Orts Escolano S, Morell GiméNez V. 2012. Autonomous growing neural gas for applications with time constraint: optimal parameter estimation. *Neural Networks* **32(4–5)**:196–208 DOI 10.1016/j.neunet.2012.02.032.

García-Rodríguez J, Angelopoulou A, Morell V, Orts S, Psarrou A, García-Chamizo JM. 2011. Fast image representation with GPU-based growing neural gas. In: Cabestany J, Rojas I, Joya G, eds. *Advances in Computational Intelligence*. Berlin: Springer, 58–65.

Girolami M. 2002. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks* **13(3)**:780–784 DOI 10.1109/TNN.2002.1000150.

Hastie T, Tibshirani R, Friedman J. 2009. *The elements of statistical learning: data mining, inference, and prediction, Springer series in statistics.* New York: Springer.

Haykin SS. 2009. *Neural networks and learning machines*. Third Edition. Upper Saddle River: Pearson Education.

Holdstein Y, Fischer A. 2008. Three-dimensional surface reconstruction using meshing growing neural gas (MGNG). *The Visual Computer* **24(4)**:295–302 DOI 10.1007/s00371-007-0202-z.

Izquierdo-Verdiguier E, Jenssen R, Gómez-Chova L, Camps-Valls G. 2015. Spectral clustering with the probabilistic cluster kernel. *Neurocomputing* **149**:1299–1304 DOI 10.1016/j.neucom.2014.08.068.

Jin R, Kou C, Liu R, Li Y. 2013. Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment. *Journal of Cloud Computing* **2(1)**:18 DOI 10.1186/2192-113X-2-18.

Khacef L, Miramond B, Barrientos D, Upegui A. 2019. Self-organizing neurons: toward brain-inspired unsupervised learning. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–9.

Khacef L, Rodriguez L, Miramond B. 2020. Improving self-organizing maps with unsupervised feature extraction. In: Yang H, Pasupa K, Leung AC-S, Kwok JT, Chan JH, King I, eds. *Neural Information Processing*. Cham: Springer.

Khan RA, Amjad RA, Kleinsteuber M. 2018. Extended affinity propagation: global discovery and local insights. *Available at http://arxiv.org/abs/1803.04459*.

Kohonen T. 1990. The self-organizing map. *Proceedings of the IEEE* **78(9)**:1464–1480 DOI 10.1109/5.58325.

Lecun Y, Bottou L, Bengio Y, Haffner P. 1998. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. 2278–2324.

Li M, Lian X, Kwok JT, Lu B. 2011. Time and space efficient spectral clustering via column sampling. In: *CVPR 2011*. 2297–2304.

Li Z, Nie F, Chang X, Nie L, Zhang H, Yang Y. 2018. Rank-constrained spectral clustering with flexible embedding. *IEEE Transactions on Neural Networks and Learning Systems* **29(12)**:6073–6082 DOI 10.1109/TNNLS.2018.2817538.

Lu C, Yan S, Lin Z. 2016. Convex sparse spectral clustering: single-view to multi-view. *IEEE Transactions on Image Processing* **25(6)**:2833–2843 DOI 10.1109/TIP.2016.2553459.

Luo P, Wu Z, Xia C, Feng L, Ma T. 2013. Co-segmentation of 3D shapes via multi-view spectral clustering. *The Visual Computer* **29(6–8)**:587–597 DOI 10.1007/s00371-013-0824-2.

MacQueen JB. 1967. Some methods for classification and analysis of multivariate observations. In: Cam LML, Neyman J, eds. *Proceedings of 5th Berkeley Symposium Mathematical Statistics and Probability.* Vol. 1. Berkeley: University of California Press, 281–297.

Mall R, Langone R, Suykens JAK. 2013. Kernel spectral clustering for big data networks. *Entropy* **15(12)**:1567–1586 DOI 10.3390/e15051567.

Martinetz T, Schulten K. 1991. A "neural-gas" network learns topologies. *Artificial Neural Networks* **I**:397–402.

Mitsyn SV, Ososkov GA. 2011. The growing neural gas and clustering of large amounts of data. *Optical Memory and Neural Networks* **20(4)**:260–270 DOI 10.3103/S1060992X11040060.

Moazzen Y, Taşdemir K. 2016. A neural gas based approximate spectral clustering ensemble. In: Merényi E, Mendenhall MJ, O'Driscoll P, eds. *Advances in Self-Organizing Maps and Learning Vector Quantization.* Cham: Springer, 85–93.

Ning C, Hongyi Z. 2016. An optimizing algorithm of non-linear k-means clustering. *International Journal of Database Theory and Application* **9(4)**:97–106 DOI 10.14257/ijdta.2016.9.4.09.

Park S, Zhao H. 2018. Spectral clustering based on learning similarity matrix. *Bioinformatics* **34(12)**:2069–2076 DOI 10.1093/bioinformatics/bty050.

Sakai T, Imiya A. 2009. Fast spectral clustering with random projection and sampling. In: Perner P, ed. *Machine Learning and Data Mining in Pattern Recognition.* Berlin: Springer, 372–384.

Shi M, Xu G. 2017. Spectral clustering using Nyström approximation for the accurate identification of cancer molecular subtypes. *Scientific Reports* **7**:4896 DOI 10.1038/s41598-017-05275-3.

Song Y, Chen W-Y, Bai H, Lin C-J, Chang E. 2008. Parallel spectral clustering. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD).* 374–389.

Sun Q, Liu H, Harada T. 2017. Online growing neural gas for anomaly detection in changing surveillance scenes. *Pattern Recognition* **64(3)**:187–201 DOI 10.1016/j.patcog.2016.09.016.

Szymański J, Dziubich T. 2017. Spectral clustering Wikipedia keyword-based search results. *Frontiers in Robotics and AI* **3**:78.

Taşdemir K. 2012. Vector quantization based approximate spectral clustering of large datasets. *Pattern Recognition* **45(8)**:3034–3044 DOI 10.1016/j.patcog.2012.02.012.

Tasdemir K, Milenov P, Tapsall B. 2011. Topology-based hierarchical clustering of self-organizing maps. *IEEE Transactions on Neural Networks* **22(3)**:474–485 DOI 10.1109/TNN.2011.2107527.

Tasdemir K, Yalçin B, Yildirim I. 2015. Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures. *Pattern Recognition* **48(4)**:1465–1477 DOI 10.1016/j.patcog.2014.10.023.

Vesanto J, Alhoniemi E. 2000. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks* **11(3)**:586–600 DOI 10.1109/72.846731.

Vojácek L, Dvorský J. 2013. Growing neural gas—a parallel approach. In: Saeed K, Chaki R, Cortesi A, Wierzchon S, eds. *Computer Information Systems and Industrial Management.* Berlin: Springer, 408–419.

von Luxburg U. 2007. A tutorial on spectral clustering. *Statistics and Computing* **17(4)**:395–416 DOI 10.1007/s11222-007-9033-z.

Wang X, Zheng X, Qin F, Zhao B. 2013. A fast spectral clustering method based on growing vector quantization for large data sets. In: Motoda H, Wu Z, Cao L, Zaiane O, Yao M, Wang W, eds. *Advanced Data Mining and Applications.* Berlin: Springer, 25–33.

Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D. 2007. Top 10 algorithms in data mining. *Knowledge and Information Systems* **14(1)**:1–37 DOI 10.1007/s10115-007-0114-2.

Yan D, Huang L, Jordan MI. 2009. Fast approximate spectral clustering. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'09.* New York: Association for Computing Machinery, 907–916.

Yu Z, You J, Han G, Li L, Wang X. 2012. Fast normalized cut algorithm based on self-organizing map. In: *2012 International Conference on Machine Learning and Cybernetics.* **4**:1376–1382.

Zhang X, You Q. 2011. An improved spectral clustering algorithm based on random walk. *Frontiers of Computer Science in China* **5(3)**:268–278 DOI 10.1007/s11704-011-0023-0.