Chapter 10

Performance Assessment of Unsupervised Clustering Algorithms Combined MDL Index

Hadeel K. Aljobouri, Hussain A. Jaber and
Ilyas Çankaya

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74506

Abstract

Best clustering analysis should be resisting the presence of outliers and be less sensitive to initialization as well as the input sequence ordering. This chapter compares the performance among three of the unsupervised clustering algorithms: neural gas (NG), growing neural gas (GNG), and robust growing neural gas (RGNG). A complete explanation of NG and GNG algorithms is presented in the next comparison with RGNG. Another comparison due to the minimum description length (MDL) criterion between RGNG used MDL value as the clustering validity index versus GNG and NG combined with MDL. Statistical estimations are applied to explain the meaning of the output results when these algorithms are fed to the synthetic 2D dataset. The techniques introduced in this chapter are designed and implemented in a simple software package using a MATLAB-based graphical user interface (GUI) tool, which allows users to interact with the clustering techniques and output data easily.

Keywords: clustering techniques, graphical user interface (GUI), growing neural gas (GNG), neural gas (NG), robust growing neural gas (RGNG)

1. Introduction

Cluster analysis [1] is a robust tool for exploring the underlining structures in data and grouping them with similar objects called clusters. Cluster analysis found applications in different fields ranging from the main task of data mining applications [2] such as scientific data exploration, spatial database applications, web analysis, marketing, medical diagnostics, computational biology, etc., to statistical data analysis that is used in many fields including machine learning, pattern recognition [3], image analysis [4], information retrieval [5], and bioinformatics [6]. There are different algorithms related to neural networks; the most popular are K-means, the self-organizing map (SOM), neural gas (NG), and growing neural gas (GNG) [7, 8].

The goal of this work is to present a comparison among neural gas (NG), growing neural gas (GNG), and robust growing neural gas (RGNG) approaches that are related to neural networks, as well as design a new simulation tool for the purpose of education and scientific research using unsupervised learning methods. Due to the difficulty in introducing these algorithms in literature, the three techniques have been presented using a simple graphical user interface (GUI) model. Alziarjawey et al. [9] introduced an application of Matlab GUI in the medical field using the ECG signal for heart rate monitoring and PQRST detection. They introduced another application by developing a software package based on GUI, which consists of two modules using many important methods derived from linear algebra [10]. Aljobouri et al. [11] designed an educational tool for biosignal processing and medical imaging using a GUI package. The user friendly package explained in this work can be used easily by: choosing any method, changing the predefined parameters for each algorithm and comparing the results. Hence, it can be used without any programming knowledge. The interested reader may find more technical details in our previous reports and publications [12, 13].

The current study is organized as follows: Section 2 provides the unsupervised clustering algorithms. Case studies are described in Section 3. Sections 4 and 5 present the experimental implementation on the synthetic dataset and clustering package design, respectively. Finally, Section 6 concludes the paper and introduces future work.

2. Unsupervised clustering algorithms

In this section, a review of the NG, GNG, and RGNG algorithms are presented. Because of the length and complexity of these algorithms, along with the mathematical model, flowcharts are designed for the three algorithms in this work in order to make it more understandable and easier to write the related codes.

2.1. NG algorithm

The NG network algorithm is a simple artificial neural network algorithm for finding optimal data representations based on reference vectors (prototype vectors). It was first introduced in 1991 [14] and is based on Kohonen's SOM [15]. Because of the dynamics of the reference vectors during the adaptation process, this algorithm was called "neural gas" that spread itself as a gas through the data space. NG is unlike other methods that consider distance as a rank like Euclidean distance, but it proposes a new way of calculating the influence of distance. Nearer prototypes in NG algorithm are more affected, but it does not depend directly on the influence of distance.

NG has been successfully applied to clustering [16], speech recognition [17], image processing [18], vector quantization, pattern recognition, topology representation, etc., [19, 20] especially where there is a problem arriving at vector quantization or data compression.

It adapts the reference vectors (prototype vectors) "w_i" without any fixed topological arrangement within the network. NG not just adapts the winner vector for a specific input vector as a single-layered soft competitive learning neural network, but also updates the residual reference vectors according to the input vector nearness using a soft-max updating rule [21]. The main advantages of NG network [22] are: (1) lower distortion error than other clustering algorithms (k-means, maximum-entropy and SOM), (2) faster assemblage due to low distortion errors, (3) submission a stochastic gradient descent on a specific energy surface.

The NG algorithm is represented by the dependence of updating strengths for c reference vectors $w_{ci}(i_0, i_1, \dots, i_{N-1})$ on their position ranking. If the input vector is presented by x, the definition of the position ranking $(w_{i0}, w_{i1}, \dots, w_{ik})$ of the reference vectors w_{ci} will be:

w_{i0} is adjacent to x.

w_{i1} is second adjacent to x

for $k = 1, 2, \dots, N-1$; $\|x - w_j\| < \|x - w_{ik}\|$, where w_{ik} is the reference vector, which has k vectors w_j.

$k_i(x, w)$ is the ranking index associated with each weight w_i.

The updating step of adjusting w_i according to a Hebb-like learning rule is given by:

$$\Delta w_i = \varepsilon(t).h_\lambda\big(k_i(x, w)\big).(x - w_i), \quad i = 1, 2, \dots, c \tag{1}$$

where:

$h_{(\cdot,\cdot)}$: deterministic function with some regularity condition imposed on it.

$\varepsilon(t) \in [0, 1]$: the learning rate (step size) that characterize the total range of the variation. This extent is represented by $\{\varepsilon(t) = \varepsilon_i.\left(\varepsilon_f/\varepsilon_i\right)^{t/(\mathrm{Max\_ter})}\}$, so Max_iter, so and t denote the maximum number of repetitions and the repetition step respectively.

$h_\lambda\big(\, k_i(v, w) \in [0, 1]$: considers the w_i within the input extent.

for $h_\lambda(k) \in [0, 1]$, the exponential form $\exp(-k/\lambda)$ was proposed [22] to obtain the best extensive result compared to other options like the Gaussian function.

λ: finds the number of reference vectors that significantly change their positions in the updating steps and usually individually decrease with the iteration step t as: $\lambda(t) = \lambda_i.\left(\lambda_f/\lambda_i\right)^{t/(\mathrm{Max\_iter})}$.

The NG algorithm is widely related to the structure of fuzzy clustering methods [23]. So, NG used the uncertainty of the relationship value $\big(h_\lambda(k_i(x, w))\big)/(C(\lambda))$to set each input vector "x" to all the reference vectors w_i $(i = 1, 2, \dots, c)$instead of using u_{ij} $(2 \le i \le c, 1 \le j \le N)$ in FCM algorithm. This algorithm is based on solving a cost function using iterative methods plus the familiarity with linear optimization methods, essentially the gradient descent method and Newton's method. Therefore, the NG cost function to optimize [22] is:

$$E_{ng} = \frac{1}{2C(\lambda)} \sum_{i=1}^{c} \int P(x)\, h_\lambda\big(k_i(x, w)\big) \left\| x - w_i \right\|^2 \tag{2}$$

with

$$C(\lambda) = \sum_{i=1}^{c} h_{\lambda}(k_i) = \sum_{k=0}^{c-1} h_{\lambda}(k) \tag{3}$$

Martinetz et al. [22, 26] introduced this cost function and proved that the updating in the Hebb-like learning rule can be derived by a stochastic gradient descent on this function. By starting with a large value of λ and reducing it slowly, a good reference vector can be obtained.

Due to the sequential learning scheme in NG algorithms and the use of the neighborhood dealing rule, NG became less sensitive to various initializations due to the sequential learning scheme and use of neighborhood cooperation rule with comparison to other clustering algorithms like k-means and FCM.

Before feeding the NG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

$\varepsilon_i, \varepsilon_f$: step size

λ_i, λ_f: decay constant

T_i, T_f: life-time

$t_{\max}$ = Max_iter = (maximal number of iterations)

Figure 1 shows the flowchart of the NG algorithm. Although the NG model has many advantages as mentioned earlier, it also has some limitations. It depends on decaying parameters that change over time; it is incapable of finding a network size and structure automatically and continue learning. Hence, based on the NG algorithm, the GNG algorithm was introduced by Fritzke [24, 25], which has an advantage over NG algorithms through its ability to modify the network topology by removing edges with its age variable. Moreover, during the growth process associated with the neighborhood updating rule, there is no need for the neighborhood sorting step [24, 25]. It has the ability to find a network size and structure automatically, and continue learning, adding units and connections, until a performance criterion is fulfilled.

2.2. GNG algorithm

In the GNG algorithm, Fritzke [24, 27] proposed changing the unit numbers (mostly increased) during SOM network with a variable topological structure [24, 25]. This growth mechanism is combined with topology formation rules using the competitive Hebbian learning (CHL) [26] and the earlier proposed growing mechanism inherited from the growing cell structures [27] to form a new model.

The GNG algorithm needs only constant parameters; it is not required to set the amount of prototypes. The main idea behind the GNG is to start with a minimal network size and insert a few new neurons and connections respectively in a growing structure by using a vector quantization until the desired characteristics of the model is fulfilled (e.g., net size, time limit, predefined number of neurons inserted, quality or some performance measure). To determine where to insert new units, local error measures are gathered during the adaptation process.

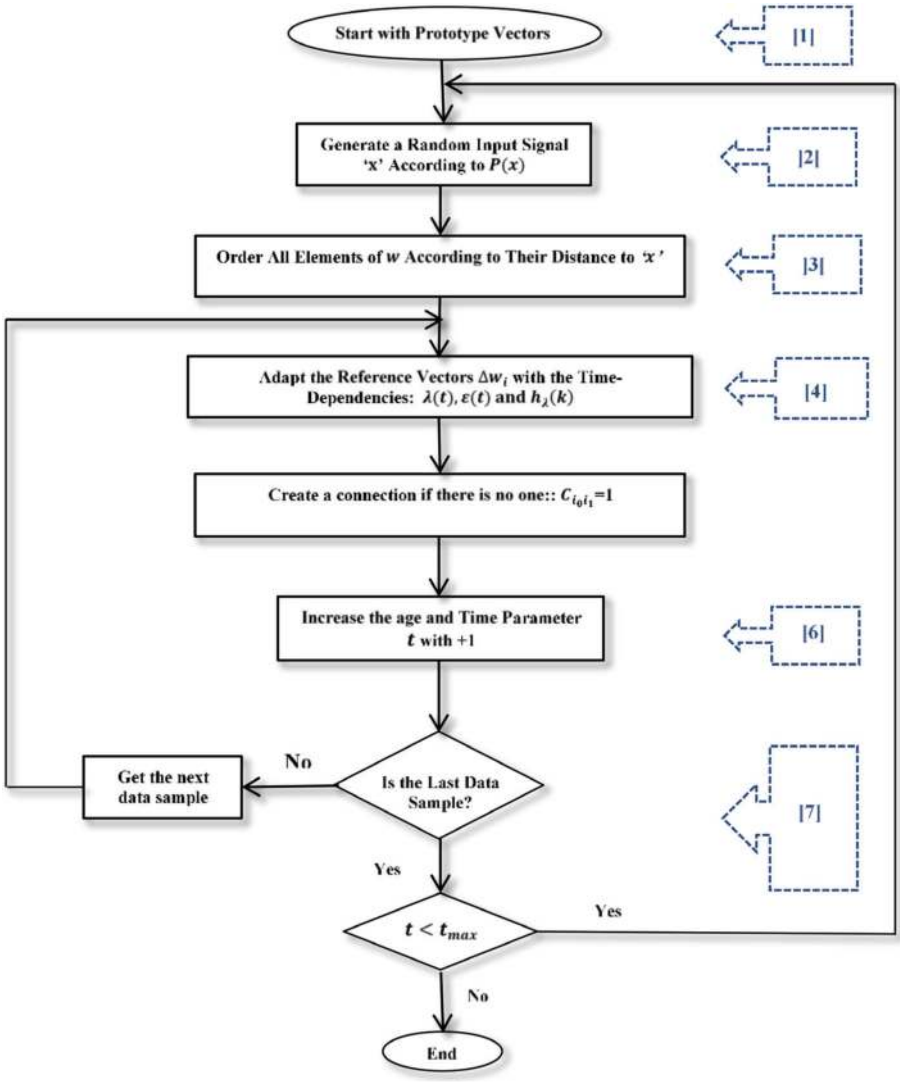

Start with Prototype Vectors
[1]
Generate a Random Input Signal 'x' According to $P(x)$
[2]
Order All Elements of w According to Their Distance to 'x'
[3]
Adapt the Reference Vectors Δw_i with the Time-Dependencies: $\lambda(t), \varepsilon(t)$ and $h_\lambda(k)$
[4]
Create a connection if there is no one:: $C_{i_0 i_1}=1$
Increase the age and Time Parameter t with +1
[6]
Is the Last Data Sample?
No
Get the next data sample
[7]
Yes
$t < t_{max}$
Yes
No
End

Figure 1. The flowchart of an NG algorithm.

Each new unit is inserted near the unit that has accumulated the highest error, and a connection between the winner and the second nearest neuron is formed using the competitive Hebbian learning algorithm.

Before feeding the GNG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

$\varepsilon_b, \varepsilon_n$: constant learning rate for the winner and its topological neighbors, respectively

λ: iteration number

α: reduction of error counter by inserting a new neuron

β: value that will reduce the overall value of the error counter every iteration step

Max_iter:: maximal number of iterations

Each reference vector $w_i, i = 1, 2, \dots, c$, has a set of edges emanating from it. It is defined to connect with its direct topological neighbors. The GNG algorithm starts by initializing a few prototype vectors (usually two) $W = \{w_1, w_2\}$ with reference vectors that are chosen randomly. New prototype vectors are successively inserted. The learning rates $\varepsilon_b, \varepsilon_n$ are used in the training procedure and a connection is formed C, $C \subset w \times w$, to the empty set: $C = \emptyset$.

The pre-specified maximum number of prototypes or neurons is set to grow as pre_numnode; and the maximum predefined training epoch Max_iter is set during each growth stage with the largest local accumulated error measure. The data set used for training is $X = \{x_1, x_2, \dots, x_N\}$. Then, the initial training epoch number is set as $m = 0$ and the iteration step in the training epoch m is set as: $t = 0$.

Figure 2 presents the flowchart of the GNG algorithm. This figure shows that nonfunctional prototypes that do not win over long time intervals may be detected by tracing the changes of an age variable associated with each edge. Hence, the GNG algorithm has an advantage against the NG algorithm through its ability to modify the network topology by removing edges with their age variable (not being refreshed for a time interval α_max) and the resultant nonfunctional prototypes. In the GNG algorithm, the growth process associated with the neighborhood updating rule used is somewhat similar to the neighborhood, decreasing procedure in NG. However, unlike the NG algorithm, there is no need for the neighborhood sorting step.

2.3. RGNG algorithm

Any robust algorithm should have the following features [28]:

1. It should achieve a good precision for the given model.
2. The performance of the given model may have few deviations from the assumptions made, but these deviations should not weaken the performance, except by a small degree.
3. The presence of large deviations from the model assumptions should not cause disaster.

If classical clustering methods are to be used as prototype based clustering algorithms, the major robustness problems are the sensitivity to initialization, the order of input vectors, and existence of many outliers, but each well executed regarding condition 1. Due to the growth scheme associated with the GNG algorithm, the algorithm faces the “dead nodes” problem. This occurs due to inappropriate initializations that led to some prototypes that may never win through the training process.

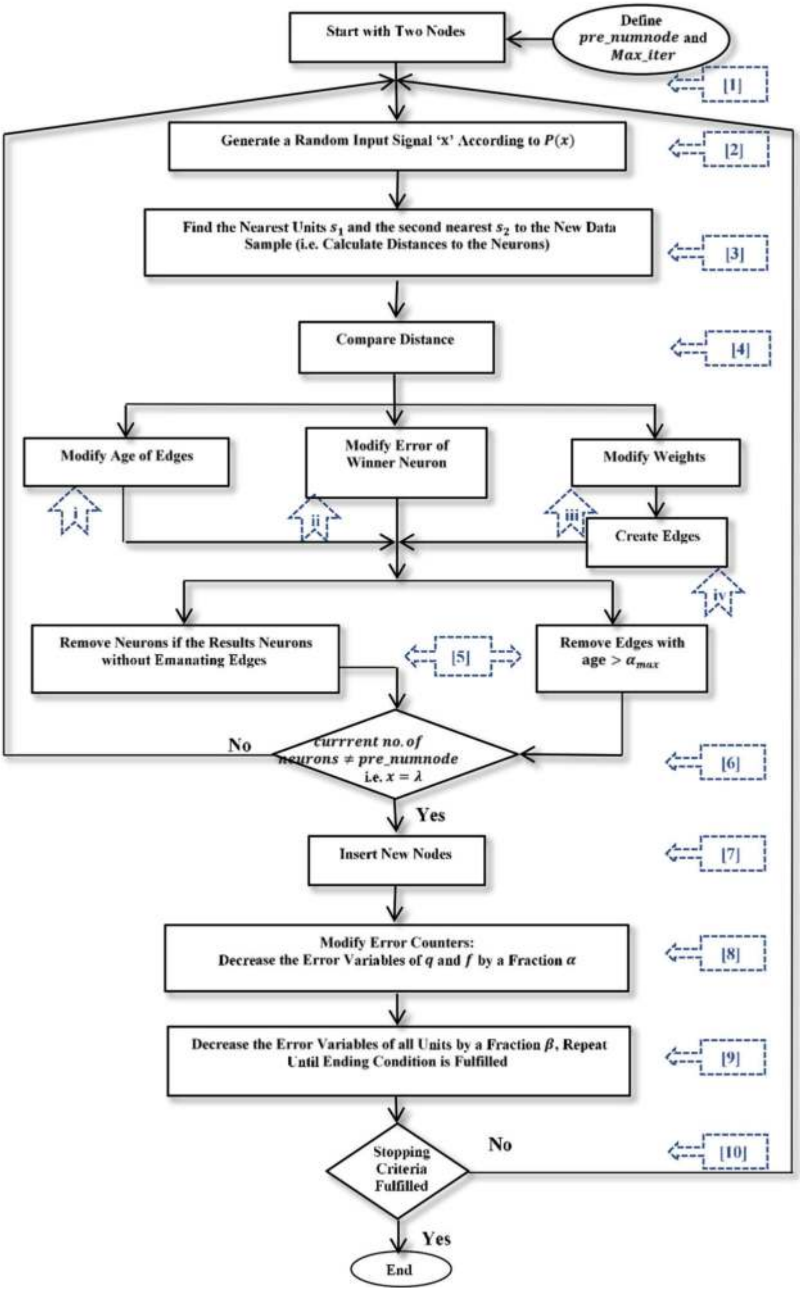

Start with Two Nodes
Define pre_numnode and Max_iter
[1]
Generate a Random Input Signal 'x' According to $P(x)$
[2]
Find the Nearest Units s_1 and the second nearest s_2 to the New Data Sample (i.e. Calculate Distances to the Neurons)
[3]
Compare Distance
[4]
Modify Age of Edges
Modify Error of Winner Neuron
Modify Weights
i
ii
iii
Create Edges
iv
Remove Neurons if the Results Neurons without Emanating Edges
[5]
Remove Edges with age > α_{max}
No
currrent no. of neurons ≠ pre_numnode i.e. $x = \lambda$
[6]
Yes
Insert New Nodes
[7]
Modify Error Counters: Decrease the Error Variables of q and f by a Fraction α
[8]
Decrease the Error Variables of all Units by a Fraction β, Repeat Until Ending Condition is Fulfilled
[9]
Stopping Criteria Fulfilled
No
[10]
Yes
End

Figure 2. The flowchart of the GNG algorithm.

Even with initialization-insensitive clustering methods, good clustering results may not be obtained if the order of the input sequence is not chosen properly.

Even with the initialization insensitive clustering methods, good clustering results may not be obtained if the order of the input sequence is not chosen properly. As well as the introduced problem gets along with the sensitivity for initialization and the order of input vectors, there also another problem attributable to the existence of many outliers. This implies the GNG network may fail to differentiate the outliers from the inliers through the original prototype updating rule when many of outliers exist in a data set. These outliers can be regarded as input vectors that different from data points belonging to the ordinary clusters (inliers).

For these limitations of the GNG algorithm, a novel robust clustering algorithm was proposed [29] within the GNG structure, namely the robust growing neural gas (RGNG) network. RGNG possesses better robustness than the original GNG algorithm because of its succession properties. It also incorporates with it several robust strategies, such as outlier resistant scheme, adaptive modulation of learning rates, and cluster repulsion method.

Therefore, compared to the GNG network, the RGNG network is insensitive to initialization, input sequence ordering, the presence of outliers, and determination of the optimal number of clusters. The minimum description length (MDL) value was used with RGNG as the clustering validity index [30, 31]. The MDL value is used to find the optimal number of clusters and their center positions corresponding to the smallest MDL. This determined automatically the optimal number of clusters by searching the extreme value of the MDL measure through the network growing process.

Before feeding the RGNG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

ε_b^l: learning rate of the winner

ε_n^l: learning rate of its topological neighbors

$\varepsilon_{bf}^l, \varepsilon_{bi}^l, \varepsilon_{nf}^l, \varepsilon_{ni}^l$: initial and final values of ε_b^l and ε_n^l

$\alpha_{\max}$: maximal age of a connection

β: mobility of the winner's neighborhood toward the input vector

k, η: parameters used to determine the MDL value

Max_iter: maximal number of iterations

The maximum number of nodes may be set to increase the pre_numnode and Max_iter and during each step with a defined number of nodes. The initial training epoch number ($m = 0$) and the iteration stage in training epoch m at $t = 0$ may also be set. Hence, the total iteration step iter during each increasing step is $\text{iter} = m.N + t$, where N is an actual number of the neuron. The dataset used for training is $X = \{x_1, x_2, \ldots, x_N\}$.

Figure 3 presents the flowchart of the RGNG algorithm.

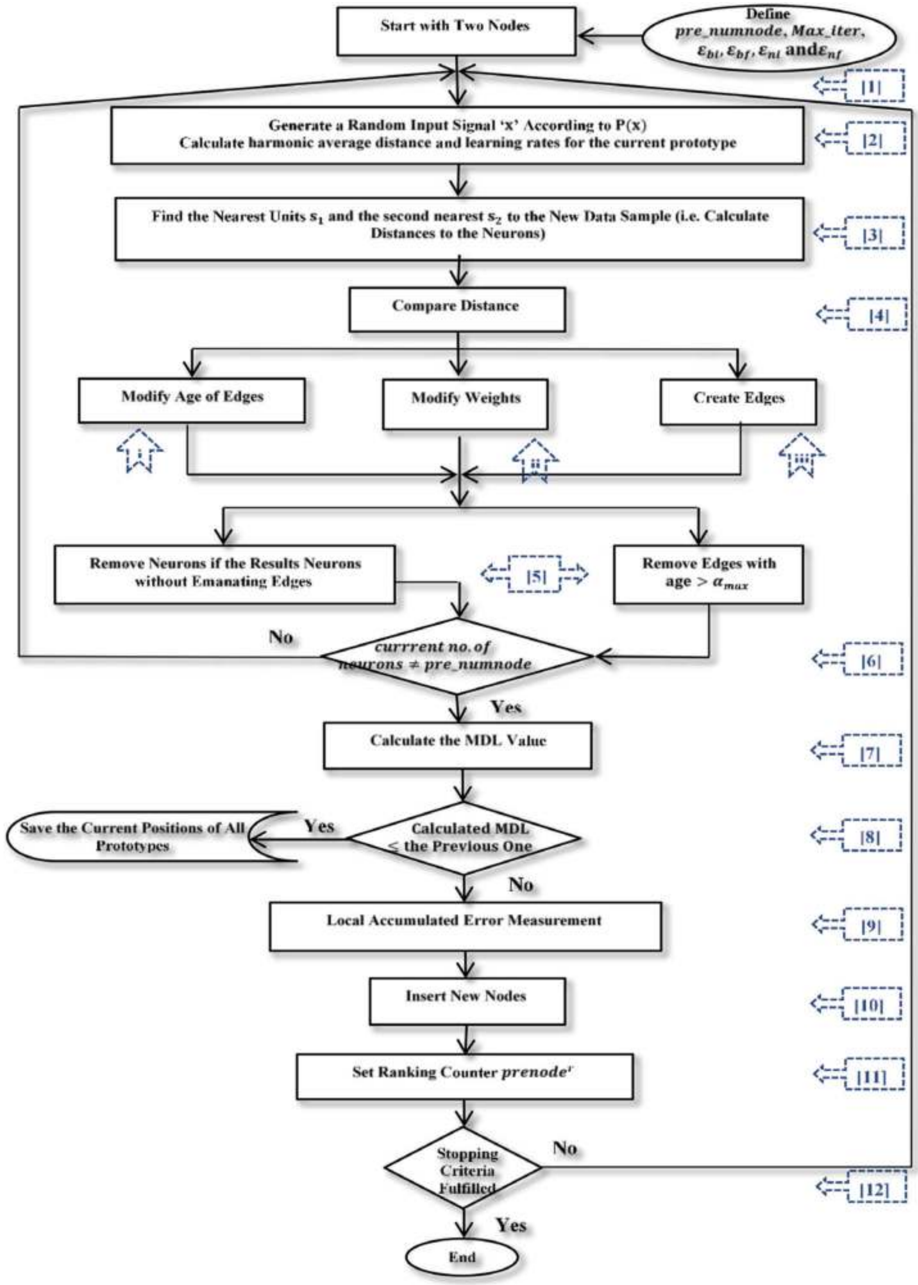

Start with Two Nodes
Define $pre\_numnode$, Max_iter, $\varepsilon_{bi}, \varepsilon_{bf}, \varepsilon_{ni}$ and ε_{nf}
[1]
Generate a Random Input Signal 'x' According to P(x)
Calculate harmonic average distance and learning rates for the current prototype
[2]
Find the Nearest Units s_1 and the second nearest s_2 to the New Data Sample (i.e. Calculate Distances to the Neurons)
[3]
Compare Distance
[4]
Modify Age of Edges
Modify Weights
Create Edges
i
ii
iii
Remove Neurons if the Results Neurons without Emanating Edges
[5]
Remove Edges with age > α_{max}
No
currrent no. of neurons ≠ pre_numnode
[6]
Yes
Calculate the MDL Value
[7]
Save the Current Positions of All Prototypes
Yes
Calculated MDL ≤ the Previous One
[8]
No
Local Accumulated Error Measurement
[9]
Insert New Nodes
[10]
Set Ranking Counter $prenode^r$
[11]
Stopping Criteria Fulfilled
No
[12]
Yes
End

Figure 3. The flowchart of the RGNG algorithm.

3. Case studies

In the presented work, the performance of the NG, GNG, and RGNG algorithms on synthetic data are described. The cases studies are carried out to compare the performance of the three approaches. The experimental results on a public synthetic dataset are presented in the next section. Comparison of different neural networks and the need for such performance parameters using statistical evaluations has been recently highlighted by a number of researchers.

There are four parameters that are used in this work to evaluate the performance of the proposed clustering technique. These performance measures are: classification rate (CR), average partition quality (PQ), minimum cluster number (MCN), and mean square error (MSE). A robust clustering technique should be less sensitive to parameter configurations and give better performance under the same parameter settings in all experiments.

In the following experiments, the parameters are fixed for each technique with typical values suggested in literature. The RGNG technique was set with the typical values provided by Qin and Suganthan [29]: $\varepsilon_{bi} = 0.1, \varepsilon_{bf} = 0.01, \varepsilon_{ni} = 0.005, \varepsilon_{nf} = 0.0005, \alpha_{\max} = 100, k = 1.3, \eta = 1 \times 10^{-4}$. GNG and NG techniques were set with the typical values provided by Fritzke [24]: $\varepsilon_b = 0.05, \varepsilon_n = 0.006, \alpha_{\max} = 100, \beta = 0.0005, \lambda = 300$ for GNG; and $\varepsilon_i = 0.5, \varepsilon_f = 0.005, \lambda_i = 10, \lambda_f = 0.01, t_{\max} = 40000$ for NG network.

Each index of the performance measures is explained in the following sections.

3.1. Classification rate

This index refers to the classification rate (CR) for the whole dataset so that each data point is classified according to its nearest prototype. CR is based on using a majority voting classifier [32] by labeling all prototypes using a simple voting mechanism. According to the proposed technique, the numbers of prototypes are small, so the resulting CR will not be high.

3.2. Partition quality

This index refers to the average partition quality (PQ) measurement, which is averaged over all the independent runs in the experiments. PQ was defined by Hamerly and Elkan [33], as:

$$\mathrm{PQ} = \frac{\sum_{i=1}^{n_{cs}} \sum_{j=1}^{n_{ct}} p\,(i, j)^2}{\sum_{i=1}^{n_{cs}} p\,(i)^2} \tag{4}$$

where:

n_{cs}: true number of classes

m_{ct}: minimum number of clusters found by the technique

$p(i, j)$: probability of a point vector in cluster j belonging to the class i

$p(i)$: class probability

The number of classes n_{cs} should equal the actual number of clusters if each natural cluster is assumed to stand for an individual class. The minimum cluster number m_{ct} can be obtained by running the techniques.

The $p(i, j)$ term represents the frequency based on the probability that a data point is labeled by clusters i and j. The $p(i, j)$ quality is normalized by the sum of true probabilities; then, squared. This statistic is related to the rand statistic for comparing partitions [34]. The PQ index is maximized when the number of clusters m_{ct} is correctly detected and induces the same partition of n_{cs}, i.e., $m_{ct} = n_{cs}$, so that all points in each cluster are the same as those in one of the natural clusters.

3.3. Minimum cluster number

The minimum cluster number (MCN) is the average number of detected clusters by the techniques. The MCN indexes the ability of the techniques to find the underlying natural clusters. During the training of the techniques and according to the MCN value, only the proposed RGNG approach can find the actual number of clusters successfully.

During the growing process, this value is defined as the number of natural clusters in which the algorithm places at least one prototype when the number of prototypes in the network reaches the actual number of clusters. Cluster numbers detected by NG and GNG during the growing process deviate from the actual value of clusters when the number of prototypes is the same as the actual number of clusters.

3.4. Mean square error

Mean square error (MSE) is another criterion used for evaluating the performance of the proposed clustering technique. The MSE value represents the mean distance between the current nearest prototypes' positions resulting from the application of the techniques and the actual cluster centers.

The average MSE value in this experiment is higher for NG and GNG techniques than the RGNG technique. This indicates that the RGNG approach achieves the best accuracy with the strongest stability among the three approaches.

4. Experimental results with synthetic data

There are six different types of 2D synthetic datasets [29, 35] which are used in this work. They are snail, screw, ring, set3, set5, and set25 dataset. **Figures 4–6** show the plots of NG, GNG, and RGNG clustering with three types of 2D synthetic datasets (screw, set5, and snail) as an example. The number of neurons are selected randomly, $N = 7$, 10, and 12.

These figures cannot clearly differentiate between each method. Hence, four parameters are used in this work to evaluate the performance of the proposed clustering techniques: CR, PQ, MCN, and MSE introduced in the previous section. For the best comparison with RGNG, MDL criterion is added to NG and GNG techniques. The training results of these techniques

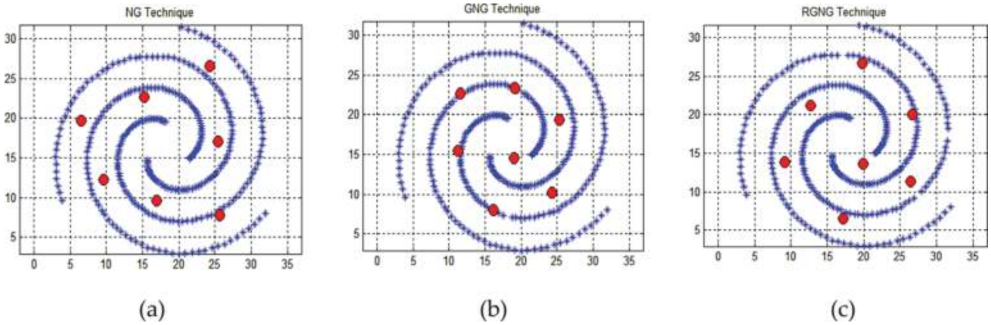


Figure 4. Clustering with screw synthetic dataset for $N = 7$, by running NG, GNG, and RGNG techniques.

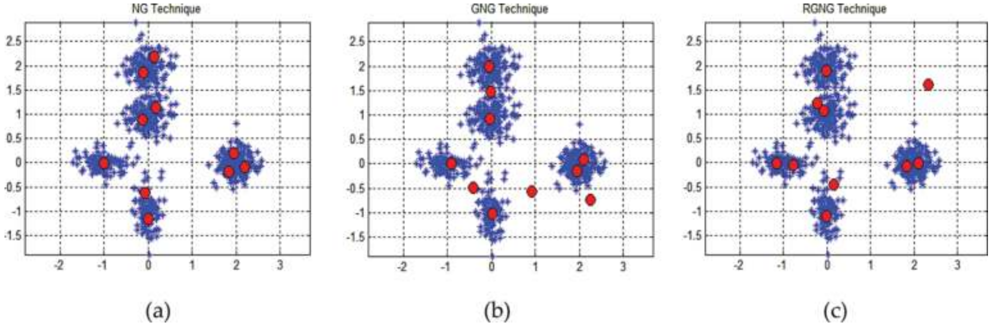


Figure 5. Clustering with set5 synthetic dataset for $N = 10$, by running NG, GNG, and RGNG techniques.

with synthetic data are shown in **Table 1**, where the number of neurons is chosen randomly as $N = 7$, 10, and 12.

According to literature [29, 36], the clustering output results introduced in **Table 1** clarified that RGNG approach is insensitive to different initializations and the presence of outliers. In

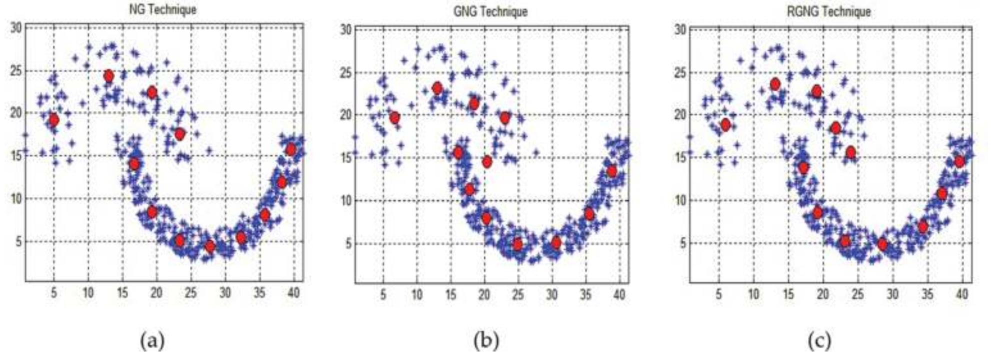


Figure 6. Clustering with snail synthetic dataset for $N = 12$, by running NG, GNG, and RGNG techniques.

Parameters	Number of neurons	NG	GNG	RGNG
CR	N = 7	0.8718	0.9686	0.9929
	N = 10	0.8514	0.9786	0.9843
	N = 12	0.8010	0.9647	0.9759
MCN	N = 7	9	8	7
	N = 10	12	11	10
	N = 12	15	14	12
PQ	N = 7	0.8990	0.9465	0.9869
	N = 10	0.8531	0.9288	0.9841
	N = 12	08279	0.9043	0.9807
MSE	N = 7	2.8032e+004	2.7608e+004	2.6493e+004
	N = 10	2.7913e+004	2.7378e+004	2.6351e+004
	N = 12	2.7703e+004	2.6940e+004	2.6188e+004

Table 1. Clustering results of synthetic data.

these techniques, the number of neurons used is small, so the CR values registered in the table are not high. In all the three clustering techniques, the number of neurons was equal to the actual cluster number. RGNG can effectively locate the actual number of clusters compared to the other two methods; NG and GNG fail with higher cluster numbers in the synthetic case.

The registered values of the MCN show that the number of detected prototypes or clusters in the RGNG technique is less than the others; which means that its ability to group data in actual number of clusters is better than the other two techniques. For example, when *N* is set to 10, the MCN value for RGNG is 10, which is less than that for NG and GNG values. The MCN value for running RGNG is equal to the number of neurons, 10, and has the same rate when compared with other *N* values; while the MCN value of running NG and GNG deviated from the actual cluster number.

Regarding the PQ value, it is noticed that the RGNG approach possesses higher PQ values than the NG and GNG techniques. For example, when *N* is set to 12, the PQ value for RGNG is 0.9807, which is higher than that of NG and GNG values. These high values of PQ indicate that the RGNG technique has a better partitioning quality with respect to the others, and finds more representative clusters.

Moreover, the RGNG method can find all the natural clusters during the growing stage with the correct number of prototypes. Hence, the MSE values are lower, which indicates that the RGNG technique has better robustness. For example, when *N* is set to 7, the MSE value for RGNG is 2.6493e + 004, which is lower than that for NG and GNG values. NG and GNG techniques may not detect all the actual clusters; hence, they yield higher MSE values.

The MDL value is one of the popular information theory evaluation measures that are used as clustering validity indexes [37]. The MDL criterion gives the ability of finding the optimal number of clusters and their center positions, corresponding to the smallest MDL value.

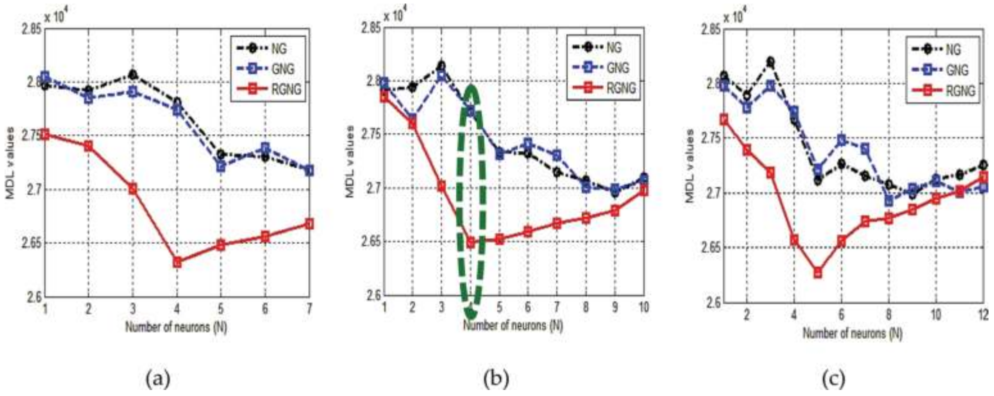


Figure 7. MDL values versus the number of clusters running the NG, GNG, and RGNG techniques on synthetic data, for: (a) $N = 7$; (b) $N = 10$; (c) $N = 12$.

The average MDL values during the growth stages are plotted versus the number of clusters or prototypes. **Figure 7** shows the curves for the NG and GNG techniques combined with the MDL criterion, as well as the RGNG approach on a synthetic dataset for different number of neurons, which are selected randomly as $N = 7$, 10, and 12. Each detected the cluster number corresponding to the MDL value.

In RGNG, the smallest MDL value was recorded on average with respect to NG and GNG combined with the MDL principle. For example, in **Figure 7 (b)**, the smallest MDL value is 2.65 that is obtained from running RGNG when N is equal to 4. While in the same $N = 4$, higher MDL value of 2.77 is recorded from running NG and GNG. From the presented figures, it is concluded that the proposed RGNG approach is insensitive to different initializations and the presence of outliers and can successfully find the actual number of clusters.

5. Prototype-based clustering package

The techniques introduced in this work are designed and implemented in a simple software package tool that allows users to interact with the clustering techniques and output data easily [13]. **Figure 8** shows the main window with the most important features of the designed prototype-based clustering software package.

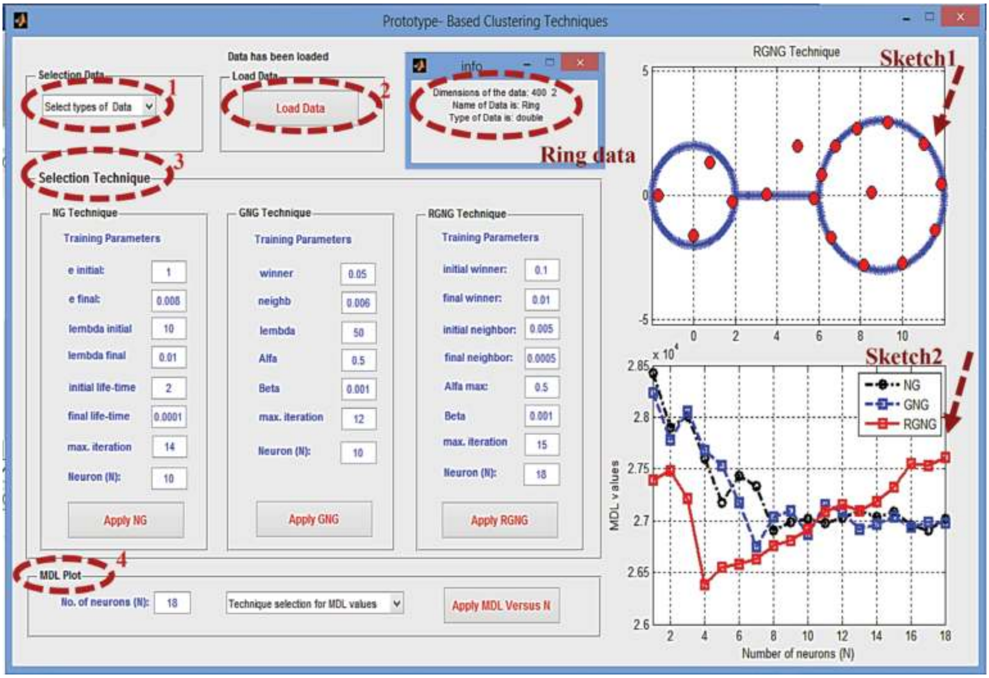
1. *Selection data*: The user can select any one type of data from the different synthetic 2D datasets in the pop-up menu. Ring data is a 2D synthetic data selected as an example in **Figure 8**.
2. *Load data*: The selected data are loaded and all information related to the selected data (“Dimension,” “Name,” and “Type of Data”) appear in the “info” window. The dimension

of the selected “Ring” data is 400x2 double. The selected data is plotted on sketch1 inside the main clustering window of **Figure 8**.

Figure 9 shows some of selected 2D synthetic datasets from the different datasets that were used in this work. Beside each plot, the information related to it is shown in the “info” window, in the left side of each plot.

3. *Selection technique*: The user can select one of the clustering techniques NG, GNG, or RGNG. The RGNG technique is selected as an example for the training in **Figure 8** with Ring data and $N = 18$, which is selected randomly.

Before clicking on “Apply NG,” “Apply GNG,” or “Apply RGNG” button, the training parameters related to each technique must be defined. As explained in Section 3, the training parameters must be set carefully within the limited range. The number of neurons (N) as well as the other parameters related to the selected technique must be defined. Another example of using the RGNG technique with Set3 dataset is shown in **Figure 10**. RGNG training parameters are set as the typical values in literature: $\varepsilon_{bi} = 0.1$, $\varepsilon_{bf} = 0.01$, $\varepsilon_{ni} = 0.005$, $\varepsilon_{nf} = 0.0005$, $\alpha_{\max} = 100$, $k = 1.3$, $\eta = 1 \times 10^{-4}$; the number of neurons (N) is chosen randomly as 14. When the algorithm’s training is



Data has been loaded

Dimensions of the data: 400 2
Name of Data is: Ring
Type of Data is: double

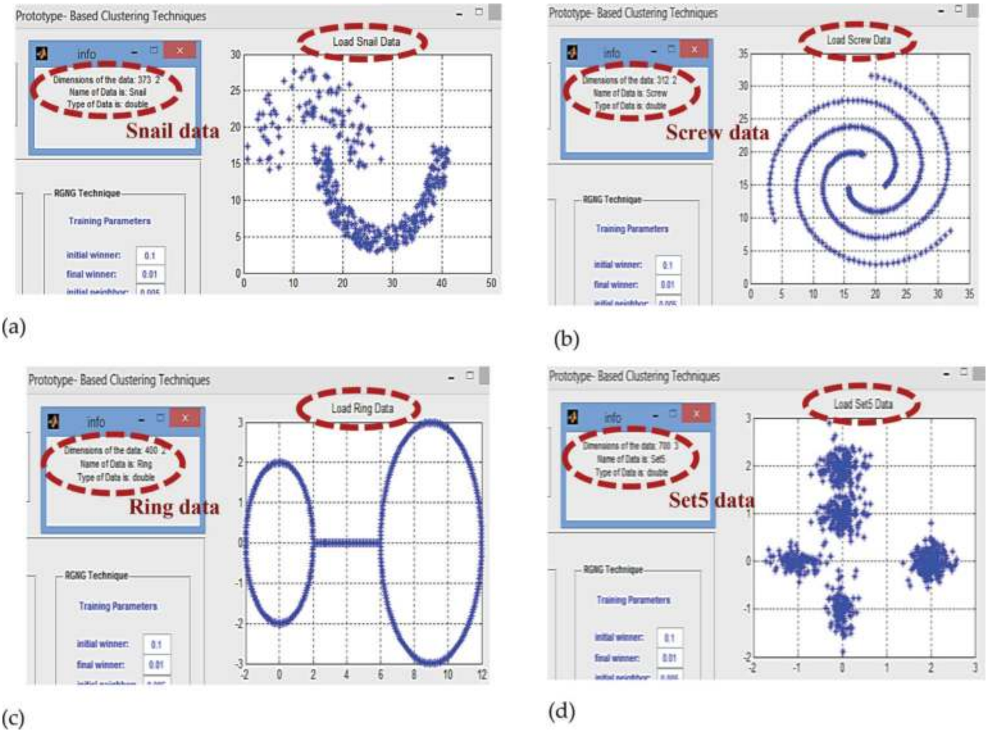
NG Technique — Training Parameters
e initial: 1
e final: 0.008
lembda initial: 10
lembda final: 0.01
initial life-time: 2
final life-time: 0.0001
max. iteration: 14
Neuron (N): 10

GNG Technique — Training Parameters
winner: 0.05
neighb: 0.006
lembda: 50
Alfa: 0.5
Beta: 0.001
max. iteration: 12
Neuron (N): 10

RGNG Technique — Training Parameters
initial winner: 0.1
final winner: 0.01
initial neighbor: 0.005
final neighbor: 0.0005
Alfa max: 0.5
Beta: 0.001
max. iteration: 15
Neuron (N): 18

MDL Plot
No. of neurons (N): 18

Figure 8. Main window of the prototype-based clustering software package.


Dimensions of the data: 373 2
Name of Data is: Snail
Type of Data is: double
initial winner: 0.1
final winner: 0.01

Dimensions of the data: 312 2
Name of Data is: Screw
Type of Data is: double
initial winner: 0.1
final winner: 0.01

Dimensions of the data: 400 2
Name of Data is: Ring
Type of Data is: double
initial winner: 0.1
final winner: 0.01

Dimensions of the data: 700 3
Name of Data is: Set5
Type of Data is: double
initial winner: 0.1
final winner: 0.01

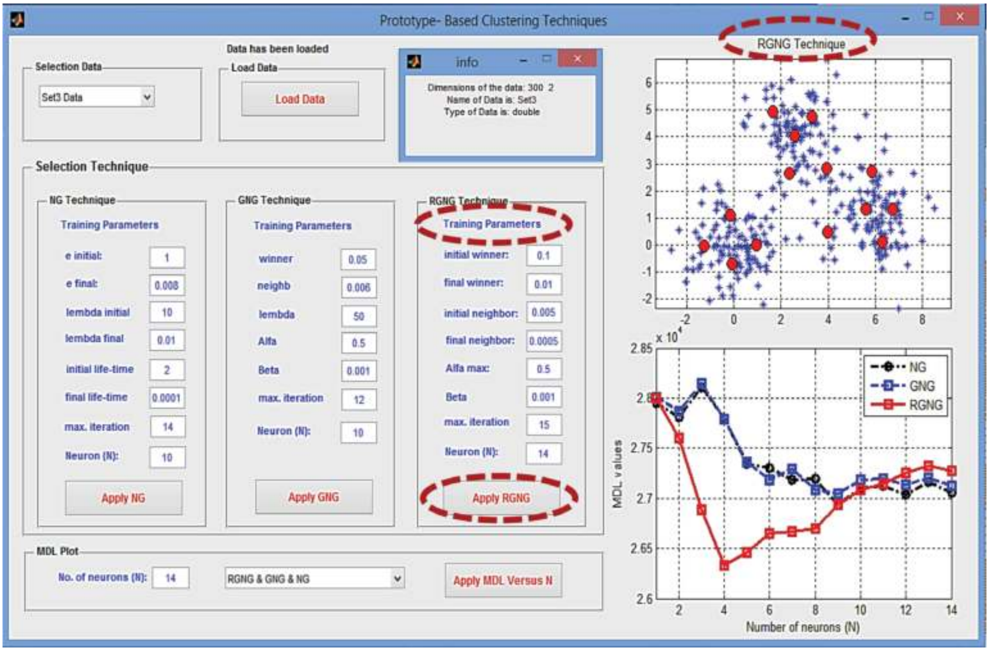
Figure 9. Different datasets with their information: (a) snail data; (b) screw data; (c) ring data; (d) Set5 data.

started, the program sketches the output running of the implemented technique as Sketch1. In Sketch1, a Set3 data is shown with firm red circles, which represent the actual cluster centers.

4. *MDL plot*: This panel is related to plotting MDL values versus the number of neurons (*N*) running the RGNG, GNG, and NG combined with MDL criterion. This panel includes three main buttons: "No. of neurons (N)," "Technique selection for MDL value," and "Apply MDL versus N" buttons, as shown in **Figure 11**.

After defining the number of neurons (*N*); one, two, or three of the training techniques have to be selected for comparing the MDL results. In the "Technique selection for MDL value" pop-up menu, there are seven selections—either show the result of each technique alone, two of them, or three of them for easy comparison. After clicking on the "Apply MDL versus N" button, the output results of MDL values are plotted with respect to the number of neurons (*N*) in Sketch2.

Figure 11 shows an example of the MDL plot, defining N = 16 and choosing "RGNG & GNG & NG" for comparing the results of the three techniques in Sketch2. For easy and best comparison between the MDL values of the three techniques, the output results sketch in the same figure.



Prototype- Based Clustering Techniques

Selection Data: Set3 Data

Data has been loaded

info
Dimensions of the data: 300 2
Name of Data is: Set3
Type of Data is: double

Selection Technique

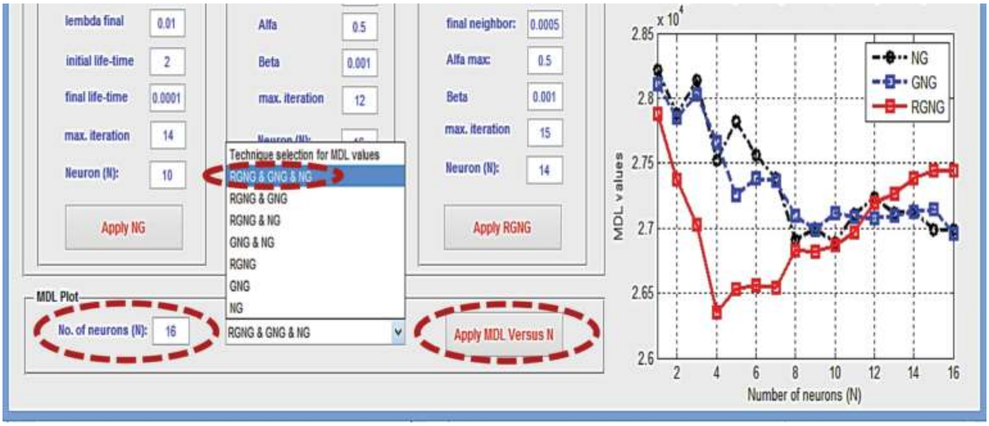
NG Technique
Training Parameters
e initial: 1
e final: 0.008
lembda initial 10
lembda final 0.01
initial life-time 2
final life-time 0.0001
max. iteration 14
Neuron (N): 10

GNG Technique
Training Parameters
winner 0.05
neighb 0.006
lembda 50
Alfa 0.5
Beta 0.001
max. iteration 12
Neuron (N): 10

RGNG Technique
Training Parameters
initial winner: 0.1
final winner: 0.01
initial neighbor: 0.005
final neighbor: 0.0005
Alfa max: 0.5
Beta 0.001
max. iteration 15
Neuron (N): 14

MDL Plot
No. of neurons (N): 14
RGNG & GNG & NG

Figure 10. RGNG clustering with Set3 data ($N = 14$).



lembda final 0.01
initial life-time 2
final life-time 0.0001
max. iteration 14
Neuron (N): 10

Alfa 0.5
Beta 0.001
max. iteration 12

final neighbor: 0.0005
Alfa max: 0.5
Beta 0.001
max. iteration 15
Neuron (N): 14

Technique selection for MDL values
RGNG & GNG & NG
RGNG & GNG
RGNG & NG
GNG & NG
RGNG
GNG
NG

MDL Plot
No. of neurons (N): 16
RGNG & GNG & NG

Figure 11. Comparison of MDL values for $N = 16$.

6. Conclusions

A simple user friendly software package is designed and implemented as an automatic clustering model for any dataset to use as part of the neural network course. NG, GNG, and RGNG algorithms are performed in the same package using a MATLAB-based graphical user interface (GUI) tool. This visual tool lets the students/ researchers visualize the desired results using plots obtained with the click of a few buttons. The performance of these algorithms on 2D synthetic datasets is reported with respect to statistical estimations to explain the meaning of the output results. These results clarified that RGNG is better than NG and GNG when considering insensitivity to initialization as well as the presence of outliers. RGNG enhances GNG to be more robust toward noisy input dataset by using MDL criteria. Hence, RGNG solves the problem of finding the optimal number of clusters with respect to NG and GNG.

For future research directions, other unsupervised or supervised clustering algorithms may be used in the laboratory experiments. Another research direction is to apply the comparison among the three clustering algorithms to real multimodal datasets in medical applications. The package results could also be shared to websites using ASP .NET, which can give facility for users by sharing applications which requires no installation of MATLAB or any special program just a Web browser.

Author details

Hadeel K. Aljobouri[1,2]*, Hussain A. Jaber[2] and Ilyas Çankaya[2]

*Address all correspondence to: hadeel_bme77@yahoo.com

1 Biomedical Engineering Department, College of Engineering, Al-Nahrain University, Baghdad, Iraq

2 Electrical and Electronics Engineering Department, Graduate School of Natural Science, Ankara Yıldırım Beyazıt University, Ankara, Turkey

References

[1] Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys (CSUR). 1999;**31**(3):264-323

[2] Berkhin P. Survey of Clustering Data Mining Techniques. Technical Report. Accrue Software Inc.; 2002

[3] Kato N, Nemoto Y. Large scale handwritten character recognition system using subspace methods. Proceeding of IEEE International Conference on Systems, Man and Cybernetics, Beijing, China, 1996. pp. 1996432-1996437

[4] Ray S, Turi RH. Determination of number of clusters in k-means clustering and application in color image segmentation. Proceeding of the Fourth International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), Calcutta, India, 1999. pp.137-143

[5] Bhatia SK, Deogun JS. Conceptual clustering in information retrieval. IEEE Transactions on System, Man, Cybernetics, Part B. 1998;**28**(3):427-436

[6] Ressom H, Wang D, Natarajan P. Adaptive double self-organizing maps for clustering gene expression profiles. Neural Networks. 2003;**16**(5-6):633-640

[7] Duda RO, Hart PE, Storck DG. Pattern Classification. New York: Wiley-Interscience; 2000

[8] Ripley BD. Pattern Recognition and Neural Networks. New York: Cambridge University Press; 1996

[9] Alziarjawey HA, Cankaya I. Heart rate monitoring and PQRST detection based on graphical user interface with Matlab. IJIEE. 2015;**5**:311-316

[10] Alziarjawey HAJ, Çamdalı Ü, Çankaya I, Aljobouri H. Design graphical user interface of linear algebra system package by using MATLAB. IJRITCC. 2016;**4**:428-433

[11] AlJobouri HK, Alziarjawey HA, Cankaya I. Biosignal Processing. Medical Imaging and fMRI (BSPMI) Software Package Based on MATLAB GUI for Education and Research. 2015;**1**:2380-8128

[12] Aljobouri HK, Çankaya I, Karal O. From biomedical signal processing techniques to fMRI Parcellation. Biosciences Biotechnology Research Asia. 2015;**12**(2):1115-1138

[13] AlJobouri HK, Jaber HA, Çankaya I. Performance evaluation of prototype-based clustering algorithms combined MDL index. Computer Applications in Engineering Education, Wiley Inc. 2017;**25**(4):642-654

[14] Martinetz T, Schulten KA. "Neural Gas" Network Learns Topologies. Artificial Neural Networks. Elsevier; 1991. pp. 397-402

[15] Kohonen T. Self-Organizing Maps. 3rd ed. Berlin: Springer; 2001

[16] Fernando C, Max C. Modification of the growing neural gas algorithm for cluster analysis. Progress in Pattern Recognition, Image Analysis and Applications, Lecture Notes in Computer Science, Springer. 2007;**4756**:684-693

[17] Curatelli F, Mayora-Iberra O. Competitive learning methods for efficient vector Quantizations in a speech recognition environment. MICAI 2000: Advances in artificial intelligence, lecture notes in computer science. Spring. 2000;**1793**:108-114

[18] Anastassia A, Alexandra P, José GR, Kenneth R. Automatic Landmarking of 2D medical shapes using the growing neural gas network. Computer vision for biomedical image applications, lecture notes in computer science. Spring. 2005;**3765**:210-219

[19] Atukorale AS, Downs T, Suganthan PN. Boosting the HONG network. Neurocomputing. 2003;**51**:75-86

[20] Winter M, Metta G, Sandini G. Neural-gas for Function Approximation: A heuristic for minimizing the local estimation error. Proceeding of International Joint Conference on Neural Network (IJCNN), Italy. 2000:535-538

[21] Haykin S. Neural Networks: A Comprehensive Foundation. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall; 1998

[22] Martinetz TM, Berkovich SG, Schulten KJ. "Neural gas" network for vector quantization and its application to time series prediction. IEEE Transactions on Neural Networks. 1993;**4**(4):558-569

[23] Bezdek JC, Keller JM, Krishnapuram R, Pal NR. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Norwell, MA: Kluwer; 1999

[24] Fritzke B. Some Competitive Learning Methods (Draft). Technique Report, Institute for Neural Computation. Bochum: Ruhr-University; 1997

[25] Fritzke B. A Growing Neural Gas Network Learns Topologies. Advances in Neural Information Processing Systems 7, Cambridge: MIT Press; 1995:625-632

[26] Martinetz TM Competitive Hebbian learning rule forms perfectly topology preserving maps. Proceedings of International Conference on Artificial Neural Networks (ICANN93), Amsterdam, The Netherlands. 1993. pp. 427-434

[27] Fritzke B. Growing cells structures—A self-organizing network for unsupervised and supervised learning. Neural Networks. 1994;**7**(9):1441-1460

[28] Huber PJ. Robust Statistics. New York: Wiley; 1981

[29] Qin AK, Suganthan PN. Robust growing neural gas algorithm with application in cluster analysis. Neural Networks, Elsevier Ltd. 2004;**17**:1135-1148

[30] Tenmoto H, Kudo M, Shimbo M. MDL-based selection of the number of components in mixture models for pattern classification. Advance in pattern recognition. Lecture Notes in Computer Science. 1998;**1451**:831-836

[31] Zemel RS. A Minimum Description Length Framework for Unsupervised Learning. Ph.D. Thesis. University of Toronto; 1994

[32] Gareth J. Majority Vote Classifiers: Theory and Applications. Ph.D. dissertation submitted to Stanford University; 1998

[33] Hamerly G, Elkan C. Learning the k in k-means. Proceeding of 17th annual conference on neural information processing systems (NIPS2003). Canada; 2003

[34] Hubert L, Arabie P. Comparing partitions. Journal of Classification. 1985;**2**:193-218

[35] Retrieved 2015, from www.yarpiz.com. 2015

[36] Qin AK, Suganthan PN. Enhanced neural gas network for prototype-based clustering. Pattern Recognition, Elsevier Ltd. 2005;**38**:1275-1288

[37] Rissanen J. A universal prior for integers and estimation by minimum description length. Annals of Statistics. 1983;**11**:416-431