Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 17 (2004) 1135–1148
Neural Networks
www.elsevier.com/locate/neunet

2004 Special Issue

Robust growing neural gas algorithm with application in cluster analysis

A.K. Qin, P.N. Suganthan*

School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Block S2, Singapore, Singapore 639798

Received 18 December 2003; accepted 3 June 2004

Abstract

We propose a novel robust clustering algorithm within the Growing Neural Gas (GNG) framework, called Robust Growing Neural Gas (RGNG) network.[1] By incorporating several robust strategies, such as outlier resistant scheme, adaptive modulation of learning rates and cluster repulsion method into the traditional GNG framework, the proposed RGNG network possesses better robustness properties. The RGNG is insensitive to initialization, input sequence ordering and the presence of outliers. Furthermore, the RGNG network can automatically determine the optimal number of clusters by seeking the extreme value of the Minimum Description Length (MDL) measure during network growing process. The resulting center positions of the optimal number of clusters represented by prototype vectors are close to the actual ones irrespective of the existence of outliers. Topology relationships among these prototypes can also be established. Experimental results have shown the superior performance of our proposed method over the original GNG incorporating MDL method, called GNG-M, in static data clustering tasks on both artificial and UCI data sets.

Keywords: Robust clustering algorithm; Robust growing neural gas; Prototypes; Outlier resistant; Minimum description length; Topology formation

1. Introduction

Cluster analysis (Jain, Murty, & Flynn, 1999) is a crucial and powerful tool for exploring and discovering the underlining structures in data by crisply or fuzzily partitioning a set of N input vectors $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N \mid \forall \mathbf{x}_i \in R^d, i=1,2,\ldots,N\}$ into c, $2\leq c\leq N$ natural groups, called clusters, such that each input vector can be assigned to each cluster with a certain degree of belongingness. Cluster analysis has found applications in diverse fields ranging from pattern recognition (Kato & Nemoto, 1996), data mining (Berkhin, 2002), computer vision (Clark et al., 1994; Frigui & Krishnapuram, 1999), and communication (Ahalt, Krishnamurty, Chen, & Melton, 1990) to information retrieval (Bhatia & Deogun, 1998) and bioinformatics (Ressom, Wang, & Natarajan, 2003).

There exist no definite categorization standards for different clustering techniques (Jain et al., 1999). Traditionally, clustering methods can be divided into hierarchical clustering and partitioning clustering. Hierarchical schemes sequentially build nested clusters with a graphical representation known as dendrogram. Partitioning methods directly assign all the data points according to some appropriate criteria, such as similarity, density, into different clusters. Our paper focuses on the prototyped-based clustering algorithm, which is the most popular class of partitioning clustering methods and can also be generalized to hierarchical methods by superposition. Hard or crisp clustering (Duda & Hart, 1973) and fuzzy clustering (Bezdek, Keller, Krishnapuram, & Pal, 1999) are often regarded as the two main branches in prototyped-based clustering. Hard clustering assigns each data point to exactly one cluster while fuzzy clustering assigns each data point to several clusters with varying degrees of memberships.

In cluster analysis tasks, the performance of a clustering algorithm may significantly depend on the correct choice of underlying cluster numbers in data sets. However, most of the traditional clustering methods, such as hard clustering and fuzzy clustering algorithms,

* Corresponding author. Tel.: +65 6790 5404; fax: +65 6792 0415.
E-mail addresses: qinkai@pmail.ntu.edu.sg (A.K. Qin), epnsugan@ntu.edu.sg (P.N. Suganthan).

[1] The Matlab codes are available from http://www.ntu.edu.sg/home/EPNSugan.

0893-6080/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2004.06.013

need to subjectively presume this crucial parameter in advance without any prior knowledge about the structure of the data sets to be tackled. In this case, only if the number of clusters is correctly specified a priori, each prototype may discover the actual positions of natural clusters while an objective function representing the sum of distances from input vectors to their corresponding prototypes is minimized. On the contrary, if the number of clusters is wrongly chosen, the final clustering results may be unsatisfactory or even meaningless. Determining the optimal number of clusters has been a challenging problem (Bischof, Leonardis, & Selb, 1999; Frigui & Krishnapuram, 1997, 1999; Hamerly & Elkan, 2003; Pelleg & Moore, 2000; Ray & Turi, 1999; Yu, 1998). Some researchers suggested initializing the clustering process with a large number of prototypes, and then employing merging or deleting schemes during the learning process to decrease prototypes to the optimal number. The exemplars of this approach are Robust Competitive Clustering Algorithm (Frigui & Krishnapuram, 1997, 1999), Robust Vector Quantization Algorithm (Bischof et al., 1999), etc. However, it is difficult in most cases to choose a reasonably large prototype number due to lack of prior knowledge about the data sets. Moreover, some threshold values utilized in these methods for merging or deleting clusters are hard to specify. Another approach to tackle this problem employs growing schemes (Hamerly & Elkan, 2003; Pelleg & Moore, 2000; Ray & Turi, 1999; Yu, 1998), which start with a few prototypes, and then split or insert new prototypes according to some criteria. Comparatively, the growing methods often require much less computation than the pruning methods. Usually a predefined maximum number of prototypes and an extreme value of a specific validation measure are used to determine the stopping of growth and the optimal number of prototypes as well as their corresponding positions. The *X*-means algorithm (Pelleg & Moore, 2000) and *G*-means algorithm (Hamerly & Elkan, 2003) are two famous representatives. However, the performance of most of these growing approaches may deteriorate significantly when data sets are contaminated by several outliers. Further, even if the actual number of clusters is detected, the obtained positions of corresponding cluster centers will be deviated significantly from the actual positions due to outliers.

It is a common truth that outliers or noisy points are likely to be brought forth into every operating step in real engineering and scientific applications. Further, the initial prototypes' positions and the ordering of the input vectors can vary significantly. If the performance of a clustering algorithm varies significantly due to changes in the initial state, input sequence ordering or the existence of outliers, the clustering results will be of little value. Hence, it is desirable to develop robust clustering algorithms which are not only capable of determining the optimal number of clusters, but also are insensitive to initialization, input ordering and presence of outliers.

Robustness of many existing clustering algorithms is not satisfactory, as they are sensitive to initializations and input orders while the final positions of prototypes can be significantly influenced by the presence of outliers. In recent years, many robust variants (Chintalapudi & Kam, 1998; Pal, Pal, & Bezdek, 1997; Wu & Yang, 2002; Yang, Wang, & Yen, 2002) of traditional clustering methods have been proposed to address the robustness issues. However, only a few of them, such as methods (Bischof et al., 1999; Frigui & Krishnapuram, 1999) that start with a large number of prototypes, claim to have the ability to detect correctly both the optimal number of clusters and their corresponding center positions in the presence of outliers.

In this paper, we present a novel Robust Growing Neural Gas (RGNG) network algorithm. By introducing the outlier resistant strategy employed in the Robust Neural Gas algorithm (Qin & Suganthan, 2004), the adaptive modulation of each prototype's learning rate and the cluster repulsion scheme used to avoid the occurrence of coincident prototypes into the Growing Neural Gas (GNG) framework (Fritzke, 1995), our RGNG algorithm can effectively overcome the outliers' influence and suppress the sensitivity to initialization and the input sequence ordering during different growth stages. The network starts with a small number of prototypes, usually 2, and gradually grows by generating new prototypes. The adaptive learning rates for winning prototypes and its direct topological neighbors are introduced to differentiate prototypes according to their insertion order, thus newly inserted prototypes are assigned with larger learning rates to find some unidentified clusters. At every growth step, Minimum Description Length (MDL) criterion (Zemel, 1994) is employed to evaluate the cluster validity according to the current number of clusters and corresponding cluster centers. Consequently, the proposed RGNG algorithm can output the optimal number of clusters and the positions of these cluster centers corresponding to the minimal MDL value. Furthermore, the topological relations among the prototype vectors can be established automatically (Martinetz & Schulten, 1994). Comparing with the original GNG algorithm incorporating MDL criterion, named GNG-M, our proposed RGNG algorithm can search more effectively for the optimal number of clusters during the network growing process, and the corresponding positions of the optimal number of clusters are much closer to the actual cluster centers with little influence by the outliers.

This paper is organized as follows. In Section 2, we retrospect the GNG algorithm. The proposed RGNG algorithm is described in Section 3. Experimental results on artificial and UCI data sets are presented in Section 4 to demonstrate the superior performance of our RGNG algorithm over the GNG-M algorithm. Finally, in Section 5 we conclude our paper with discussion and some possible future directions.

2. Review of growing neural gas algorithm

The Neural Gas (NG) network algorithm (Martinetz, Berkovich, & Schulten, 1993) has been successfully applied to clustering, vector quantization, pattern recognition and topology representation, etc. (Atukorale, Downs, & Suganthan, 2003; Winter, Metta, & Sandini, 2000). It is similar to Kohonen's Self-Organizing Feature Map (Kohonen, 2001) but adapts the reference vectors (prototype vectors) $\mathbf{w}_i$ without any fixed topological arrangement within the network. As a single-layered soft competitive learning neural network, it not only adjusts the winner vector for a certain input vector but also updates the remaining reference vectors according to their proximities to this input vector by using a soft-max updating rule (Haykin, 1998). Three main advantages of NG model are (Martinetz et al., 1993): (1) faster convergence to low distortion errors, (2) lower distortion error than that resulting from k-means clustering, maximum-entropy clustering and Kohonen's Self-Organizing Feature Map algorithm, (3) obeying a stochastic gradient descent on an explicit energy surface.

In the NG algorithm, the updating strengths for c reference vectors $\mathbf{w}_i$, $i=1,2,\ldots,c$, depend on their positions in the 'neighborhood ranking' list. When an input vector $\mathbf{x}$ is presented, we determine the neighborhood ranking $(\mathbf{w}_{i_0}, \mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_{c-1}})$ of the reference vectors $\mathbf{w}_i$, $i=1,2,\ldots,c$, with $\mathbf{w}_{i_0}$ being the closest to $\mathbf{x}$, $\mathbf{w}_{i_1}$ being second closest to $\mathbf{x}$, and $\mathbf{w}_{i_k}$, $k=1,2,\ldots,c-1$ being the reference vector for which there are k vectors $\mathbf{w}_j$ with $\|\mathbf{x}-\mathbf{w}_j\| < \|\mathbf{x}-\mathbf{w}_{i_k}\|$. The ranking index associated with each weight $\mathbf{w}_i$ is denoted by $k_i(\mathbf{x},\mathbf{w})$. Assuming a Hebb-like rule (Haykin, 1998), the updating step for adjusting $\mathbf{w}_i$ is given by

$$\Delta\mathbf{w}_i = \varepsilon(t)\cdot h_\lambda(k_i(\mathbf{x},\mathbf{w}))\cdot(\mathbf{x}-\mathbf{w}_i), \quad i=1,2,\ldots,c. \tag{1}$$

The learning rate $\varepsilon(t)\in[0,1]$ describes the overall extent of modification and usually takes an exponential decreasing form as $\varepsilon(t)=\varepsilon_i\cdot(\varepsilon_f/\varepsilon_i)^{t/(Maxiteration)}$, where t and $Maxiteration$ stand for the iteration step t and the maximum number of iterations, and $h_\lambda(k_i(\mathbf{v},\mathbf{w}))\in[0,1]$ accounts for the topological arrangement of the $\mathbf{w}_i$ within the input space. The exponential form $\exp(-k/\lambda)$ was suggested in Martinetz et al. (1993) for $h_\lambda(k)\in[0,1]$ to obtain the best overall result, comparing with other choices like the Gaussian function. The parameter λ determines the number of reference vectors significantly changing their positions in the updating steps and usually monotonically decreases with the iteration step t as $\lambda(t)=\lambda_i\cdot(\lambda_f/\lambda_i)^{t/(Maxiteration)}$.

Regarded as one of the partitioning clustering algorithms, NG Algorithm is closely related to the framework of fuzzy clustering methods (Bezdek et al., 1999). Instead of the fuzzy membership u_{ij}, $2\le i\le c$, $1\le j\le N$ employed in fuzzy c-means algorithm (Bezdek et al., 1999), NG utilizes the uncertainty of belongingness value $(h_\lambda(k_i(\mathbf{x},\mathbf{w})))/(C(\lambda))$ to assign each input vector $\mathbf{x}$ to all the reference vectors $\mathbf{w}_i$, $i=1,2,\ldots,c$. Therefore, its cost function can be expressed as Martinetz et al. (1993):

$$E_{ng} = \frac{1}{2C(\lambda)}\sum_{i=1}^{c}\int p(\mathbf{x})h_\lambda(k_i(\mathbf{x},\mathbf{w}))\|\mathbf{x}-\mathbf{w}_i\|^2 d\mathbf{x}, \quad \text{with } C(\lambda)=\sum_{i=1}^{c}h_\lambda(k_i)=\sum_{k=0}^{c-1}h_\lambda(k). \tag{2}$$

The updating rule (1) can thus be derived by the stochastic gradient descent on this cost function as mathematically proven in Martinetz et al. (1993). To obtain good results for the set of reference vectors, we start with a large λ value and decrease it gradually during the updating procedure and correspondingly the algorithm transits from a soft model to the hard one. By using this deterministic annealing like strategy, we expect good local minimum of the cost function to emerge slowly, thereby efficiently preventing final results from getting trapped into inferior sub-optimal states. In addition, comparing with the batch mode partitioning clustering algorithms such as k-means and Fuzzy c-means, NG algorithm is seldom sensitive to different initializations due to the sequential learning scheme and use of neighborhood cooperation rule. The detailed implementation of NG algorithm is presented in Fritzke (1997).

Originating from the NG algorithm, Fritzke proposed an incremental self-organizing network with a variable topological structure, known as Growing Neural Gas (GNG) algorithm (Fritzke, 1995, 1997). It combines the growing mechanism inherited from the Growing Cell Structures (Fritzke, 1994) with topology formation rules by the Competitive Hebbian Learning scheme (Martinetz, 1993; Martinetz & Schulten, 1994). For each reference vector $\mathbf{w}_i$, $i=1,2,\ldots,c$, a set of edges emanating from it is defined to connect with its direct topological neighbors. The GNG algorithm starts with a few prototype vectors (usually two) and new prototype vectors are successively inserted, after a predefined number of training epochs Max_iter, near the prototypes featuring with the largest local accumulated error measure. Due to this gradual insertion strategy, the positions of the initial few prototypes may have little impact on the clustering performance. This growing procedure will stop when a pre-specified maximum number of prototypes or a specified performance measure is reached. In GNG, at the presence of an input vector $\mathbf{x}$, the prototype's updating is only operated on the winning prototype $\mathbf{w}_{s_1}$ and its direct topological neighbors $\mathbf{w}_i$, $\forall i\in N_{s_1}$, where N_{s_1} is the set of direct topological neighbors of the prototype $\mathbf{w}_{s_1}$ that are connected by an edge with $\mathbf{w}_{s_1}$. Furthermore, the updating strengths are different for the winning prototype and its topological neighbors while remaining constant over time. The updating rule of GNG algorithm is expressed as

follows:

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b \cdot (\mathbf{x} - \mathbf{w}_{s_1}),\ \Delta \mathbf{w}_i = \varepsilon_n \cdot (\mathbf{x} - \mathbf{w}_i),\ \forall i \in N_{s_1}. \quad (3)$$

Here, ε_b and ε_n represent the constant learning rate for the winner and its topological neighbors, respectively. The GNG algorithm can detect the inactive prototypes, which do not win during a long time interval, by tracing the changes of an age variable associated with each edge. Thus, it is able to modify the network topology by removing edges with its age variable not being refreshed for a time interval $\alpha_{\max}$ and by removing the resultant inactive prototypes. Unlike the NG algorithm, neighborhood ranking step is no longer needed since the growth process combined with the local neighborhood updating rule used in GNG is somewhat equivalent to the neighborhood decreasing procedure in NG. The complete implementation of the GNG algorithm is summarized in Table 1.

3. Robust growing neural gas algorithm

Robustness is an important characteristic of good clustering algorithms. According to Huber (Dave & Krishnapuram, 1997): a robust algorithm should possess the following properties: (1) it should achieve a reasonably good accuracy at the assumed model; (2) small deviations from the model assumptions should impair the performance only by a small amount; (3) larger deviations from the model assumption should not cause a catastrophe. Traditional clustering methods, such as prototype based clustering algorithms, often perform well under the condition (1), and the main robustness issues associated with them are the sensitivity to initialization, the order of input vectors and presence of a large number of outliers.

The GNG algorithm may seldom suffer from the 'dead nodes' problem (which means that some prototypes may never win during the training procedure because of improper initializations) because it employs the growth scheme, sequential learning strategy and soft-max updating

Table 1
Implementation of the Growing Neural Gas (GNG) algorithm

Initialize a set of prototype vectors (usually 2) $\mathbf{W}=\{\mathbf{w}_1,\mathbf{w}_2\}$ randomly, the learning rates $\varepsilon_b,\varepsilon_n$ used in the training procedure and a connection set $\mathbf{C}$, $\mathbf{C}\subset\mathbf{W}\times\mathbf{W}$ to an empty set: $\mathbf{C}=\phi$. Set the maximum number of prototypes to grow as *pre_numnode* and the maximum training epoch *Max_iter* during each growth stage with a certain number of prototypes. Set the initial training epoch number: $m=0$ and the iteration step in training epoch m: $t=0$. Hence, the total iteration step iter during each growth stage is: iter$=m\cdot N+t$. The data set used for training is $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}$.

(a) While the current number of prototypes$\leq$*pre_numnode* and some predefined performance measure, if exists, is not satisfied

(b) For $m=0$ to *Max_iter*-1

- Set $t=1$, thus iter$=m\cdot N+t$.
- Set *trainingset*$=\mathbf{X}$, i.e. include all input vectors into trainingset.

(c) If *trainingset* is not empty

- Draw randomly an input vector $\mathbf{x}_t^m$, at the iteration step t in training epoch m, from the *trainingset*
- Determine the winner s_1 and the second nearest prototype $s_2(s_1,s_2\in\mathbf{W})$ by $s_1=\text{argmin}_{i\subset\mathbf{W}}\|\mathbf{x}_t^m-\mathbf{w}_i^{\text{iter}}\|$ and $s_2=\text{argmin}_{i\subset\mathbf{W}\setminus\{s_1\}}\|\mathbf{x}_t^m-\mathbf{w}_i^{\text{iter}}\|$, where iter$=m\cdot N+t$.
- If a connection between s_1 and s_2 does not exist already, create it: $\mathbf{C}=\mathbf{C}\cup\{(s_1,s_2)\}$. Set the *age* of the connection between s_1 and s_2 to 0, i.e. $\text{age}_{(s_1,s_2)}=0$.
- Add the squared distance between input vector and the winner to a local accumulated error variable: $\Delta\mathbf{E}_{s_1}=\|\mathbf{x}_t^m-\mathbf{w}_{s_1}\|^2$.
- Adapt the reference vectors of the winner and its direct topological neighbors by fractions ε_b and ε_n, respectively, of the total distance to the input vector: $\Delta\mathbf{w}_{s_1}=\varepsilon_b\cdot(\mathbf{x}-\mathbf{w}_{s_1})$, $\Delta\mathbf{w}_i=\varepsilon_n\cdot(\mathbf{x}-\mathbf{w}_i)$, $\forall i\in N_{s_1}$, where N_{s_1} is the set of direct topological neighbors of s_1.
- Increment the *age* of all edges emanating from s_1: $\text{age}_{(s_1,i)}=\text{age}_{(s_1,i)}+1$, $\forall i\in N_{s_1}$
- Remove edges with *age* values larger than $\alpha_{\max}$. If this results in prototypes having no more emanating edges, remove those prototypes as well
- Increment the iteration step t by 1 and remove the used vector $\mathbf{x}_t^m$ from the set *trainingset*

(c) End If

(b) End For

(d) If the current number of prototypes$\neq$*pre_numnode*

- Determine the prototype q with the maximum accumulated error: $q=\text{argmax}_{i\in\mathbf{W}}\mathbf{E}_i$.
- Determine, among the neighbors of q, the prototype f with the maximum accumulated error: $f=\arg\max_{i\in N_q}\mathbf{E}_i$.
- Add a new prototype r to the network and interpolate its reference vector from q and f: $\mathbf{w}=\mathbf{w}\cup\{\mathbf{w}_r\}$, $\mathbf{w}_r=(\mathbf{w}_q+\mathbf{w}_f)/2$
- Insert edges connecting the new prototype r with prototypes q and f, respectively, and remove the original edge between q and f: $\mathbf{C}=\mathbf{C}\cup\{(r,q),(r,f)\}$, $\mathbf{C}=\mathbf{C}\setminus\{(q,f)\}$
- Decrease the error values of q and f by a fraction α: $\Delta\mathbf{E}_q=-\alpha\cdot\mathbf{E}_q$, $\Delta\mathbf{E}_f=-\alpha\cdot\mathbf{E}_f$
- Interpolate the error value of r from q and f: $\mathbf{E}_r=(\mathbf{E}_q+\mathbf{E}_f)/2$.
- Decrease the error values of all prototypes: $\Delta\mathbf{E}_i=-\text{beta}\cdot\mathbf{E}_i$, $\forall i\in\mathbf{W}$.

(d) End If

(a) End While

rule. Actually, the initialization problem is implicitly converted to the input sequence ordering issue for the sequential learning method. Therefore, if the order of the input sequence is not chosen properly, good clustering results may not be obtained even by the initialization insensitive methods. With respect to clustering, outliers can be regarded as input vectors, which are substantially dissimilar to data points belonging to the natural clusters, namely inliers. Hence, it is unreasonable to assign them to any clusters with a high certainty of belongingness value. However, if a large number of outliers exist in a data set, the original prototype updating rule in the GNG may fail to differentiate the outliers from the inliers, thus the prototypes may be significantly attracted towards these sequentially presented outliers.

By inheriting the merits of the original GNG algorithm while overcoming the robustness problems associated with it, we have devised a RGNG algorithm. Unlike the present GNG variants, the proposed RGNG algorithm is insensitive to initialization, input sequence ordering and the presence of many outliers during the training at every growth stage. By utilizing the MDL value as the clustering validity index, we can determine the optimal number of clusters and their center positions, corresponding to the smallest MDL value.

3.1. Outlier resistant strategy

In order to overcome the influence of outliers in the NG algorithm, we recently proposed a Robust Neural Gas (RNG) algorithm (Qin & Suganthan, 2004), which has exhibited good robust performance in static clustering tasks with a fixed number of prototypes in noisy environment Since the GNG algorithm can be viewed as being composed of a series of NG methods with different number of prototypes, the outlier resistant strategy devised in the RNG algorithm can be incorporated in the GNG to robustify the training procedure in noisy environment. By considering each growth step individually, we describe the strategy employed to resist outliers as follows.

By retrospection of the updating rule (3) employed in the original GNG algorithm, we find this formula is inherently fragile in noisy environment and sensitive to the order of input vectors. To clearly explain the reasons, we reformulate expression (3) as:

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b \cdot \|\mathbf{x} - \mathbf{w}_{s_1}\| \cdot \frac{(\mathbf{x} - \mathbf{w}_{s_1})}{\|\mathbf{x} - \mathbf{w}_{s_1}\|},$$
$$\Delta \mathbf{w}_i = \varepsilon_n \cdot \|\mathbf{x} - \mathbf{w}_i\| \cdot \frac{(\mathbf{x} - \mathbf{w}_i)}{\|\mathbf{x} - \mathbf{w}_i\|}, \quad \forall\, i \in N_{s_1}. \tag{4}$$

According to the formula (4), if an outlier $\mathbf{x}_O$ is presented to the GNG network, the amplitude $\|\mathbf{x}_O - \mathbf{w}_k\|$, $\forall k \in N_{s_1} \cup \{s_1\}$ generated along the unit direction $(\mathbf{x}_O - \mathbf{w}_k)/\|\mathbf{x}_O - \mathbf{w}_k\|$ will be large. Hence, the outlier can significantly influence the updating of prototypes. In addition, when outliers are located at different positions in the input sequence at a certain growth stage, the updating strength from these outliers will differ according to input orderings and thereby destabilizing the final results. For instance, if most of the outliers are presented in the beginning, the updating strength from them to all prototypes will be strong such that the set of reference vectors under this growth stage may plunge into inferior local minima. Ideally, outliers should have little impact on the prototypes' updating procedure in order to obtain the correct cluster center positions. Therefore, the absolute distance information $\|\mathbf{x} - \mathbf{w}_k\|$, $\forall k \in N_{s_1} \cup \{s_1\}$ should be modulated during the updating to identify and eliminate the influence from outliers.

In order to enhance the robustness of the updating rule (4), we modify this rule to a new form as:

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b \cdot \sigma_{s_1}(\text{iter}) \cdot \frac{(\mathbf{x} - \mathbf{w}_{s_1})}{\|\mathbf{x} - \mathbf{w}_{s_1}\|},$$
$$\Delta \mathbf{w}_i = \varepsilon_n \sigma_i(\text{iter}) \frac{(\mathbf{x} - \mathbf{w}_i)}{\|\mathbf{x} - \mathbf{w}_i\|}, \quad \forall\, i \in N_{s_1}. \tag{5}$$

In this new formula, we first substitute the absolute distance $\|\mathbf{x} - \mathbf{w}_k\|$, $\forall k \in N_{s_1} \cup \{s_1\}$ in (4) with parameter $\sigma_k(\text{iter})$, $\forall k \in N_{s_1} \cup \{s_1\}$, which is used to limit the force amplitude caused by an outlier $\mathbf{x}_O$ to $\mathbf{w}_k$. Here, parameter *iter* represents the training iteration step during a certain growth stage and when the number of prototypes is increased, iter has to be reset to one. Assuming that *Max_iter* training epochs are used to train the network under one growth stage and the learning process is at the iteration t, $1 \le t \le N$ of the training epoch m, $0 \le m \le \text{Max\_iter} - 1$ (i.e. the total iteration step at a certain growth stage is $\text{iter} = m \cdot N + t$), the parameter $\sigma_k(\text{iter})$ is defined as follows

$$\sigma_k(\text{iter}) = \sigma_k^m(t) = \begin{cases} d_k^m(t) & \text{if } \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\| \ge d_k^m(t-1) \\ \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\| & \text{if } \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\| < d_k^m(t-1) \end{cases}, \tag{6}$$

with

$$d_k^m(t) = \begin{cases} \left\{ \frac{1}{2} \left[\frac{1}{d_k^m(t-1)} + \frac{1}{\|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\|} \right] \right\}^{-1} & \text{if } \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\| \ge d_k^m(t-1) \\ \frac{1}{2}[d_k^m(t-1) + \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\|] & \text{if } \|\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}\| < d_k^m(t-1) \end{cases} \tag{7}$$

and

$$d_k^m(0) = \left[\frac{1}{N} \sum_{j=1}^{N} \frac{1}{\|\mathbf{x}_j - \mathbf{w}_k^{mN}\|} \right]^{-1}, \tag{8}$$

where N, $\mathbf{x}_t^m$ and $\mathbf{w}_k^{\text{iter}}$ represent the number of input vectors, the input vector presented at the iteration t of the training

epoch m and the reference vector k at the total iteration step iter during a certain growth stage, respectively. The term $d_k^m(t)$ serves as the restricting distance for prototype $\mathbf{w}_k$, $k \in N_{s_1} \cup \{s_1\}$, which includes both historical and current distance information and can be employed to limit the large absolute distances due to possible outliers. The initial restricting distance $d_k^m(0)$ is defined by Eq. (8) as the harmonic average of the distances from all input vectors $\mathbf{x}_j$, $j=1,2,\ldots,N$ to each prototype $\mathbf{w}_k^{m\cdot N}$,, $k=1,2,\ldots,c$ before the first iteration of training epoch m.

The meaning of the term $\sigma_k(\text{iter})$ can be explained as follows. Under a specific growth stage, before the first iteration step of each training epoch m, i.e. iter$=m\cdot N$, we calculate by Eq. (8) the harmonic average distance $d_k^m(0)$ for each prototype $\mathbf{w}_k$. During the training epoch m, when iteration step t is increased by 1, we average the absolute distances $\mathbf{x}_t^m - \mathbf{w}_k^{\text{iter}}$ with the historical restricting distance $d_k^m(t-1)$ to constitute the current restricting distance $d_k^m(t)$, as in Eq. (7). Here, Eq. (7) indicates that if the current absolute distance is greater or equal to the historical restricting distance we take the harmonic average of them, while if the current absolute distance is smaller, we take the arithmetic average of them. In this way, large absolute distance between $\mathbf{x}_t^m$ and prototypes $\mathbf{w}_k^{\text{iter}}$, $\forall k \in N_{s_1} \cup \{s_1\}$ can be suppressed. Eq. (6) shows that if the current absolute distance is larger or equal to the historical restricting distance, we take the current restricting distance as $\sigma_k(\text{iter})$, and otherwise the current absolute distance will be used. By combining the historical movement distance information with the current movement distance information for each prototype, our RGNG algorithm can gradually suppress the influence from possible outliers during the training at each growth stage and consequently the input sequence ordering problem can be significantly weakened.

Compared with Eq. (3), the newly derived updating rule does not change the gradient descent direction while heuristically and adaptively adjusts the amplitude of gradient descent (Haykin, 2001) along its original direction in order to resist the influence from the outliers. As a result, final positions of all prototypes can be correctly obtained with reduced influence from the outliers.

3.2. Adaptive learning rates and cluster repulsion scheme

With the above outlier resistant strategy, the training process at each growth stage with a certain number of prototypes is insensitive to the presence of outliers Hence, the existing prototypes are able to find meaningful regions in the data set. When a new prototype is inserted, we expect this new prototype to find an unidentified cluster in the data set while already existing prototypes should refine the cluster centers around them. However, in the original GNG algorithm, the newly inserted prototype and the existing prototypes have the same learning rates ε_b and ε_n such that both of them possess the same updating strength for input vectors from an unidentified region. In fact, during training, the current prototypes may have approximately found some clusters and they should gradually refine their positions towards the actual cluster centers. In contrast, the newly inserted prototypes should avoid finding the same clusters as the existing prototypes and try to find some new clusters. To realize this idea, we introduce the adaptive learning rates ε_b and ε_n, which take different values for prototypes inserted in different order, and meanwhile make them decrease monotonically with the increment of prototypes, i.e.

$$\varepsilon_b^l = \varepsilon_{bi}^l (\varepsilon_{bf}^l/\varepsilon_{bi}^l)^{prenode^l/pre\_numnode} \quad \text{and } \varepsilon_n^l = \varepsilon_{ni}^l (\varepsilon_{nf}^l/\varepsilon_{ni}^l)^{prenode^l/pre\_numnode}, \quad l = 1,\ldots,c \tag{9}$$

where ε_{bi}^l, ε_{bf}^l, ε_{ni}^l, ε_{ni}^l, *pre_numnode* and c stand for the initial and final values of ε_b^l and ε_n^l, which correspond to prototype l, the predefined maximum number of prototypes and the current number of prototypes in the network, respectively. Parameter prenode^l defines the ranking counter for prototype l, which is set to zero for a newly inserted prototype and incremented by 1 for this prototype after one new prototype is inserted. In this way, the 'older' the prototypes, the closer their learning rates ε_b^l and ε_n^l are towards the final values. Here, we choose the initial values to be 10 times the final values for these two learning rates. In this manner, the newly generated prototype can possess higher degree of adaptation to exploit the new regions in the data set and the existing prototypes will keep refining their positions.

However, there exists another problem in that, two prototypes may simultaneously find same clusters during the training procedure because an input vector may attract both the winner and its direct topological neighbors towards it and thus the neighboring prototypes may possibly find the same cluster. To address this prototype coincident problem (Timm, Borgelt, Döring, & Kruse, 2001), we introduce a mutual repulsion scheme for prototypes within the local neighborhood. Given an input vector $\mathbf{x}$, the corresponding winner and its direct topological neighbors can be detected. Then a repulsive force along the direction from the winner to its topological neighbors will be added to the updating rule of these neighbor prototypes. The amplitudes of the repulsive forces are defined as an integral multiple of the average distance between the winner and all its direct topological neighbors. In addition, an exponential term is employed to judge whether one neighboring prototype is very close to the winner and the exponential term can weaken the repulsive force when the neighboring prototype is far away from the winner. We incorporate the above idea into the neighbor prototypes' updating rule (5) and modified it as follows

$$\begin{aligned} \Delta\mathbf{w}_{s_1} &= \varepsilon_b^{s_1}\sigma_{s_1}(\text{iter})\frac{(\mathbf{x}-\mathbf{w}_{s_1})}{\|\mathbf{x}-\mathbf{w}_{s_1}\|} \\ \Delta\mathbf{w}_i &= \varepsilon_n^i\sigma_i(\text{iter})\frac{(\mathbf{x}-\mathbf{w}_i)}{\|\mathbf{x}-\mathbf{w}_i\|} + \exp\left(-\frac{d_{s_1 i}}{\varsigma}\right) \\ &\quad \times \beta\frac{\sum_i d_{s_1 i}}{|N_{s_1}|}\frac{(\mathbf{w}_i-\mathbf{w}_{s_1})}{\|\mathbf{w}_i-\mathbf{w}_{s_1}\|}, \quad \forall\, i \in N_{s_1}, \end{aligned} \tag{10}$$

where β and $|N_{s_1}|$ stand for the integral multiplier to define the amplitude of the repulsive force and the cardinality of the topological neighborhood of winner s_1, respectively. The parameter ς controls the weakening effect in terms of the distance between winner s_1 and its neighbors. In our following experiments, we choose values 0.1 and 2 for ς and β, and according to the preliminary experiments, the clustering performance is not very sensitive to the selection of these two parameter values. By utilizing the updating step (10) instead of (5), the newly inserted prototype can exert its ability to discover a new cluster region and the prototype coincident problem (Timm et al., 2001) can be successfully solved.

3.3. Determination of the optimal number of clusters

In clustering tasks, determining the optimal number of clusters remains an open problem. Most traditional clustering algorithms need to predefine the number of clusters. However, due to lack of knowledge about the data set to be clustered, a badly assumed number may lead to unsatisfactory results. In recent years, many cluster validity measures have been proposed to tackle this issue. Among them, information theoretical methods, such as Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC) and Minimum Message Length (MML), etc. (Pelleg & Moore, 2000; Ray & Turi, 1999; Yu, 1998), have been successfully applied to obtain good performance. In our RGNG algorithm, we employ the Minimum Description Length (MDL) criterion (Tenmoto, Kudo, & Shimbo, 1998; Zemel, 1994), one of the famous information theory evaluation indices, to serve as the cluster validity measure.

Minimum Description Length principle, which was originally proposed by Rissanen (1989) as a model selection criterion, has been widely applied in the field of neural networks (Bischof et al., 1999; Tenmoto et al., 1998; Zemel, 1994) to evaluate the network's ability to describe a given data set by balancing the complexity and the capability of the network. Here, we can accomplish the task of determining the optimal cluster numbers by finding a certain number of prototype vectors that can minimize the length of description of a set $\mathbf{V}$ of N data samples. Due to the existence of outliers, we can divide the data set $\mathbf{V}$ into two subsets $\mathbf{I}$ and $\mathbf{O}$, which are composed of inliers and outliers, respectively. Accordingly, we formulate the expression of MDL criterion as follows:

$$\mathrm{MDL}(\mathbf{V},\mathbf{W}) = \mathrm{mod}\, L(\mathbf{I},\mathbf{W}) + \mathrm{error}\, L(\mathbf{I},\mathbf{W}) + \mathrm{mod}\, L(\mathbf{O}). \tag{11}$$

$\mathbf{W}$ represent the set of c reference vectors $\mathbf{W}=\{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_c\}$. We evaluate the complexity of the entire model by the term mod $L(\mathbf{I},\mathbf{W})$, and the length of encoding it is given by the sum of the two terms: (1) the length of encoding all prototypes, denoted by $L(\mathbf{W})$. (2) The length of encoding all the indices of $\mathbf{I}$, denoted by $L(\mathbf{I}(\mathbf{W}))$. The encoding length error $L(\mathbf{I},\mathbf{W})$ represents residual errors generated by describing all inlier data points $\mathbf{I}$ with prototype set $\mathbf{W}$. The description length of the outlier set $\mathbf{O}$, denoted by mod $L(\mathbf{O})$, is usually encoded in the same way as the prototype vectors. The model's capability of describing the whole data set $\mathbf{V}=\mathbf{I}+\mathbf{O}$ can be reflected by the last two terms in Eq. (11). We instantiate the calculation of MDL value in our approach as in Bischof et al. (1999)

$$\begin{aligned}\mathrm{MDL}(\mathbf{V},\mathbf{W}) &= cK + N\log_2 c \\ &\quad + \kappa\sum_{i=1}^{c}\sum_{\mathbf{x}\in S_i}\sum_{k=1}^{d}\max\left(\log_2\left(\frac{\|\mathbf{x}_k-\mathbf{w}_{ik}\|}{\eta}\right),1\right) \\ &\quad + |\mathbf{O}|K,\end{aligned} \tag{12}$$

where c, d, η, S_i and $|\mathbf{O}|$ represent the current number of prototypes, the dimension of input vectors, the resolution of the data source, the inlier receptive field of prototype $\mathbf{w}_i$ and the cardinality of the outlier set, respectively. Here, K is the number of bits needed for encoding a single data vector and we usually calculate this value according to the average value range of input vectors and data accuracy η: $K=\lceil\log_2(\mathrm{range}/\eta)\rceil$, and for the index term $L(\mathbf{I}(\mathbf{W}))$ we adopt a fixed length encoding scheme (Bischof et al., 1999) where each inlier point $\mathbf{x}\in\mathbf{I}$ is encoded with $\log_2 c$ bits and thus $L(\mathbf{I}(\mathbf{W}))=N\log_2 c$. We assume that the length of encoding the error term error $L(\mathbf{I},\mathbf{W})$ is proportional to its magnitude, and that data accuracy is η in all d dimensions. If the error term $(\mathbf{x}-\mathbf{w}_i)$ is independent along each of d dimensions, it can be encoded separately along every dimension as

$$L(\mathbf{x}-\mathbf{w}_i) = \sum_{k=1}^{d}\max(\log_2(\|\mathbf{x}_k-\mathbf{w}_{ik}\|/\eta),1),$$

therefore, the total error encoding length is the sum of error lengths in each receptive field S_i of prototype $\mathbf{w}_i$, i.e.

$$\begin{aligned}\mathit{error}L(\mathbf{I},\mathbf{W}) &= \sum_{i=1}^{c}\sum_{\mathbf{x}\in S_i}L(\mathbf{x}-\mathbf{w}_i) \\ &= \sum_{i=1}^{c}\sum_{\mathbf{x}\in S_i}\sum_{k=1}^{d}\max(\log_2(\|\mathbf{x}_k-\mathbf{w}_{ik}\|/\eta),1).\end{aligned}$$

Parameter κ is used to balance the contribution of model complexity mod $L(\mathbf{I},\mathbf{W})$ and model efficiency error $L(\mathbf{I},\mathbf{W})$ in the calculation of MDL value. In our clustering tasks, usually $\kappa>1$ is selected.

To complete the calculation of MDL value in Eq. (12) at each growth stage, outliers corresponding to the current number of prototypes need to be detected. According to the MDL principle, outliers can be determined in the following manner (Bischof et al., 1999): a data point is judged as an outlier if the encoding length of this point by the index of its nearest prototype vector and the produced quantization error is larger than the length of encoding this data point itself.

This idea can be instantiated by checking the change in the coding length when we move a data vector $\mathbf{x} \in S_i$ from the inlier set $\mathbf{I}$ to the outlier set $\mathbf{O}$

$$\begin{aligned}\Delta L &= \{L((\mathbf{I}-\{\mathbf{x}\})(\mathbf{W}))+K\}-\{L(\mathbf{I}(\mathbf{W}))+L(\mathbf{x}-\mathbf{w}_i)\}-\varphi_{S_i}K \\ &= \{(N-1)\log_2(c-\varphi_{S_i})+K\} \\ &\quad -\left\{N\log_2 c+\sum_{k=1}^{d}\max(\log_2(\|\mathbf{x}_k-\mathbf{w}_{ik}\|/\eta),1)\right\}-\varphi_{S_i}K,\end{aligned} \tag{13}$$

where φ_{S_i} equals 1 if $\mathbf{x}$ is the only vector in the receptive field S_i before movement, otherwise it equals 0. If the value change ΔL in (13) is negative, $\mathbf{x}$ can be regarded as an outlier since encoding it alone has a smaller length than encoding it with a prototype vector. In this way, outliers can be detected by computing (13) at the end of each growth stage. It is worthy of noting that during the training procedure, when the number of prototypes grows to the actual number of clusters in the data set, if the positions of these prototypes are attracted far away from the actual cluster centers by outliers, the resultant MDL value may be larger than the MDL value obtained with the correct positions of prototypes. In this situation, the minimum value of MDL may not be obtained with the correct number of prototypes. This explains why the GNG-M algorithm may not successfully find the optimal number of prototypes and their correct positions, while our RGNG algorithm is able to.

3.4. Implementation of the RGNG algorithm

Our RGNG algorithm starts with a few prototypes (usually two), and then some training epochs are employed to train the existing number of prototypes by updating rule (10). Due to the incorporation of the outlier resistant strategy, adaptive learning rates modulation and the cluster repulsion scheme, the RGNG algorithm is robust in the presence of outliers. After a certain number of training epochs for current prototypes, a new prototype is inserted near the existing prototype with the largest local accumulated error. In noisy environment, this accumulated error measure for each prototype may not be the simple sum of quantization error based on Euclidean metric since outliers may influence the accumulated quantization error of their nearest prototypes. In our GNG algorithm, the local accumulated error of prototype $\mathbf{w}_i$ is expressed as

$$\begin{aligned}&\sum_{x\in S_i}\exp\left(-\frac{\|\mathbf{x}-\mathbf{w}_i\|}{\text{harmdist}^i}\right)\|\mathbf{x}-\mathbf{w}_i\|, \\ &\text{with harmdist}^i = \left[\frac{1}{|S_i|}\sum_{x\in S_i}\frac{1}{\|\mathbf{x}-\mathbf{w}_i\|}\right]^{-1},\end{aligned} \tag{14}$$

where S_i is the receptive field of prototype $\mathbf{w}_i$ and term harmdist^i is defined as the harmonic average distance from all data points in S_i to the prototype vector $\mathbf{w}_i$. Therefore, large accumulated error measure caused by outliers can be effectively weakened. As training proceeds, the MDL value is calculated by Eq. (12) at the completion of each growth stage until the predefined maximum number of prototypes is reached. Finally, the optimal cluster number and the corresponding cluster center positions can be obtained according to the smallest MDL value.

The complete RGNG algorithm is summarized in Table 2. According to the preliminary experiments, the proposed algorithm is less sensitive to the predetermined parameters, such as ε_{bi}, ε_{bf}, ε_{ni}, ε_{nf}, *Max_iter* and *pre_numnode*. Therefore, in Section 4, all the experiments are conducted using the same parameter values. In addition, in order to avoid too small distance values, we add a small positive number $\varepsilon = 1\times10^{-5}$ whenever we calculate distances.

4. Experimental results

In this section, we test the robust performance of our proposed RGNG algorithm on some artificial and UCI data sets. Compared to the GNG-M algorithm, which incorporates the MDL principle into the original GNG approach in the same way as the RGNG does, the RGNG algorithm shows superior performance.

Clearly, a good robust clustering algorithm should be less sensitive to the parameter configurations and perform well under the same parameter settings in all experiments. In the following experiments, the parameters in the GNG-M algorithm were set as the typical value in Fritzke (1997): $\varepsilon_b = 0.05$, $\varepsilon_n = 0.006$, $\alpha_{\max} = 100$, $\kappa = 1.3$, $\eta = 1\times10^{-4}$. And the parameter set employed in our RGNG algorithm was chosen as follows: $\varepsilon_{bi} = 0.1$, $\varepsilon_{bf} = 0.01$, $\varepsilon_{ni} = 0.005$, $\varepsilon_{nf} = 0.0005$, $\alpha_{\max} = 100$, $\kappa = 1.3$, $\eta = 1\times10^{-4}$. The maximum number of clusters *pre_numnode* is the same for both the GNG-M and RGNG algorithms, and is chosen according to the scale of the clustering tasks.

We independently run each of these two clustering algorithms 10 times. The initial states, chosen as two prototypes, are different for each of the 10 runs. The detected optimal number of clusters and some other performance measures are used to evaluate the performance of the clustering algorithms. Here, we apply the unpaired *t*-test to demonstrate the statistical significance between the performance measures obtained by the two algorithms RGNG and GNG-M, which is indicated by the *p*-values in Tables 3–5.

4.1. Synthetic data

To analyze the cluster discovery performance especially in noisy environment, we designed two 2D data sets and applied the clustering algorithms on them.

The first data set D_1 is shown in Fig. 1a. Five clusters S_1, S_2, S_3, S_4 and S_5 are generated according to different

Table 2
Implementation of the Robust Growing Neural Gas (RGNG) algorithm

Initialize a set of prototype vectors (usually 2) $\mathbf{W}=\{\mathbf{w}_1,\mathbf{w}_2\}$ randomly, the learning rates ε_{bi}, ε_{bf}, ε_{ni}, ε_{nf} used in the training procedure, the ranking counter variable prenode^l for each current prototype l to 0, a connection set $\mathbf{C}$, $\mathbf{C}\subset\mathbf{W}\times\mathbf{W}$ to an empty set: $\mathbf{C}=\phi$ and κ, η used to calculate the MDL value. Set the maximum number of prototypes to grow as *pre_numnode* and the maximum training epoch *Max_iter* during each growth stage with a certain prototype number. Set the initial training epoch number: $m=0$ and the iteration step in training epoch m: $t=0$. Hence, the total iteration step iter during each growth stage is: iter$=m\cdot N+t$. The data set used for training is $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N,\}$.

(a) While the current number of prototypes$\leq$*pre_numnode* and some predefined performance measure, if exists, is not satisfied

(b) For $m=0$ to *Max_iter*-1

- Calculate the harmonic average distance value $d_m^i(0)$ for each prototype $\mathbf{w}_i$, $\forall i\in\mathbf{W}$, w.r.t. its current position by Eq. (8)
- Calculate the learning rates for the current prototype l, when it serves as a winner or its topological neighbors, respectively, by Eq. (9). The learning rates remain same in the following training for the prototype l, before the new prototype is inserted.
- Set $t=1$, thus iter$=m\cdot N+t$.
- Set *trainingset*$=\mathbf{X}$, i.e. include all input vectors into *trainingset*.

(c) If *trainingset* is not empty

- Draw randomly an input vector $\mathbf{x}_t^m$, at the iteration step t in training epoch m, from the *trainingset*.
- Determine the winner s_1 and the second nearest prototype $s_2(s_1,s_2\in\mathbf{W})$ by $s_1=\text{argmin}_{i\subset\mathbf{w}}\|x_t^m-w_i^{\text{iter}}\|$, and $s_2=\text{argmin}_{i\subset\mathbf{W}\setminus\{s_1\}}\|\mathbf{x}_t^m-\mathbf{w}_i^{\text{iter}}\|$, where iter$=m\cdot N+t$.
- If a connection between s_1 and s_2 does not exist already, create it: $\mathbf{C}=\mathbf{C}\cup\{(s_1,s_2)\}$. Set the *age* of the connection between s_1 and s_2 to 0, i.e. $\text{age}_{(s_1,s_2)}=0$.
- Adapt the reference vectors of the winner and its direct topological neighbors according to Eqs. (6), (7), (8) and (10).
- Increment the *age* of all edges emanating from s_1: $\text{age}_{(s_1,i)}=\text{age}_{(s_1,i)}+1$, $\forall i\in N_{s_1}$
- Remove edges with *age* values larger than $\alpha_{\max}$. If this results in some prototypes having no more emanating edges, remove those prototypes as well.
- Increment the iteration step t by 1 and remove the used vector $\mathbf{x}_t^m$ from the set *trainingset*

(c) End If

(b) End For

- Calculate the MDL value according to the current prototypes' positions by Eqs. (12) and (13)
- Save the current positions of all prototypes, if the calculated MDL value is smaller than the previous one
- Determine the local accumulated error measure for each of the current prototypes according to Eq. (14)
- Determine the prototype q with the maximum accumulated error: $q=\text{argmax}_{i\in\mathbf{W}}\mathbf{E}_i$
- Determine, among the neighbors of q, the prototype f with the maximum accumulated error: $f=\arg\max_{i\in N_q}\mathbf{E}_i$.
- Add a new prototype r to the network and interpolate its reference vector from q and f: $\mathbf{w}=\mathbf{w}\cup\{\mathbf{w}_r\}$, $\mathbf{w}_r=(2\cdot\mathbf{w}_q+\mathbf{w}_f/3)$
- Insert edges connecting the new prototype r with prototypes q and f, and remove the original edge between q and f: $\mathbf{C}=\mathbf{C}\cup\{(r,q),(r,f)\}$, $\mathbf{C}=\mathbf{C}\setminus\{(q,f)\}$
- Set the ranking counter prenode^r of the newly inserted prototype r to 0 and for other existing prototypes, increment their corresponding counter variables by 1, i.e. prenode$^r=0$, prenode$^l=$prenode$^l+1$, $\forall l\in\mathbf{W}\setminus\{r\}$.

(a) End While

Gaussian distributions with centers at [0,1], [0,2], [−1,0], [1,0] and [0,−1]. Among them, S_1, S_2 and S_3 have circular shapes; S_4 and S_5 have elliptical shapes. The number of sample points for these five clusters are 150, 150, 200, 100 and 100, respectively. Fig. 1b shows the noisy version of data set D_1, in which 84 outliers are added. Fig. 2a shows the second multimodal type data set D_2. This data set has $k=25$ clusters arranged in a 5×5 grid in two dimensions. Each cluster generates 50 data points from its own Gaussian distribution, for a total of $N=1250$ data points. We added 104 noisy points to contaminate this data set as shown in Fig. 2b. In Figs. 1a,b and 2a,b solid black crosses represent the actual cluster centers. Note that, in noisy versions of data sets D_1 and D_2, four outliers with their coordinates at (10,0), (10,0), (20,0) and (20,0) are not visible in Figs. 1b and 2b because of the display effect.

The following performance measures are employed to evaluate the clustering results. Since the actual number of

Table 3
Clustering results of GNG-M and RGNG algorithms on data set D_1

Data sets	Methods	Opt_Clut	*PQ*	No_Cluts	MSE
Clean D_1	GNG-M	16.8±3.1	0.414±0.177	4.5±0.5	0.150±0.004
	RGNG	5.0±0.0	0.989±0.001	5.0±0.0	0.013±0.004
	p-value	0.000	0.000	0.005	0.000
Noisy D_1	GNG-M	10.7±1.5	0.485±0.173	1.8±1.1	1.805±0.777
	RGNG	5.0±0.0	0.755±0.031	5.0±0.0	0.016±0.005
	p-value	0.000	0.000	0.000	0.000

Table 4
Clustering results of GNG-M and RGNG algorithms on multimodal data set D_2

Data sets	Methods	Opt_Clut	*PQ*	No_Cluts	MSE
Clean D_2	GNG-M	29.1±2.6	0.957±0.019	18.3±2.0	0.082±0.124
	RGNG	25.0±0.0	1.000±0.000	25.0±0.0	0.003±0.001
	p-value	0.000	0.000	0.000	0.059
Noisy D_2	GNG-M	32.8±9.0	0.851±0.050	8.8±1.3	4.798±2.294
	RGNG	25.0±0.0	1.000±0.000	25.0±0.0	0.004±0.001
	p-value	0.013	0.000	0.000	0.000

clusters is known a priori, we use the average number of clusters detected by the algorithms over 10 runs, denoted by *opt_num*, to display the algorithms' ability to find the underlying natural clusters. The second performance measure we employed is the average partition quality (*PQ*), which is defined in Hamerly and Elkan (2003) as

$$PQ = \sum_{i=1}^{n_{class}} \sum_{j=1}^{n_{clust}} p(i,j)^2 / \sum_{i=1}^{n_{class}} p(i)^2$$

and averaged over 10 independent runs. Here we assume that each natural cluster stands for an individual class, thus the number of classes n_{class} should equal the actual number of clusters. The optimal cluster number obtained by running the algorithms is denoted as n_{clust}, and terms $p(i,j)$ and $p(i)$ represent the frequency based probability of a point vector in cluster j belonging to class i, and the class probability, respectively. This evaluation index is maximized when the number of clusters n_{clust} is correctly detected, i.e. $n_{\text{clust}}=n_{\text{class}}$, and all points in each cluster are the same as those in one of the natural clusters. To demonstrate the robustness of our proposed RGNG algorithm, some indices are utilized to evaluate the intermediate clustering results when the number of prototypes in the network reaches the actual number of clusters during the growing process: *No_Cluts* is defined as the number of natural clusters in which the algorithm placed at least one prototype; Mean Square Error (*MSE*) is the average distance between the actual cluster centers and the current prototypes' positions resulting from the application of the algorithms. These two measures are averaged over 10 runs and can reveal the robustness of the algorithms when training with the actual number of clusters in noisy environment during the growing process. The calculation of these performance measures on contaminated data sets will only use the inliers regardless of the added outliers, and this operation will highlight the algorithm's performance in finding the actual clusters.

Fig. 1c–f plot the average MDL values over 10 runs, which is obtained at the end of each growth stage by the GNG-M and RGNG algorithms on the clean and noisy data set D_1, respectively. The maximum number of prototypes for growth is predefined as 20. It is clear that our RGNG algorithm can successfully find the actual number of clusters and is insensitive to different initializations and the presence of outliers, while the GNG-M algorithm fails with wrong cluster numbers on both clean and noisy version of data set D_1. The other performance measures used to evaluate these two algorithms on data set D_1 are shown in Table 3. We can observe that our RGNG algorithm yields higher *PQ* values than the GNG-M algorithm does in both clean and noisy cases, indicating better partitioning quality of the RGNG algorithm. Meanwhile, at the growth stage with the correct number prototypes, our method can find all the natural clusters in both clean and noisy data set D_1 over each of the 10 runs and obtain lower MSE values, which shows improved robustness of our algorithm. It is worth noting that the calculation of the MDL value at each growth stage depends heavily on the current positions of the prototypes, since the encoding length of quantization errors and the detected outliers will differ significantly according to the obtained cluster centers' position. For GNG-M algorithm, we can obviously discover that when the total number of prototypes is 5 (Table 3), it may not detect all the actual clusters and possess a higher MSE value. In noisy situations, the prototypes will deviate significantly from the actual cluster center positions at the growth stage with the correct number of prototypes, which can be observed from the last two measures in Table 3 for GNG-M. However, after some prototypes are attracted near the outliers, the other prototypes will be able to find the actual clusters. These facts indicate that the GNG-M algorithm is highly sensitive to the presence of outliers. This explains why GNG-M finally results in a larger number of clusters.

The plots of the MDL values versus the number of prototypes resulting from the GNG-M and RGNG algorithms on multimodal data set D_2 and its noisy version are shown in Fig. 2c–f. Here, we predefine the maximum number of prototypes as 50. Some clustering performance indices of these two algorithms are described in Table 4. Compared with our RGNG algorithm, the GNG-M algorithm shows

Table 5
Clustering results of GNG-M and RGNG algorithms on data set wine

Data sets		Methods	Opt_Clut	*PQ*
WINE	Clean *D*2	GNG-M	3.5±0.7	0.804±0.085
		RGNG	3.0±0.0	0.875±0.031
		p-value	0.037	0.022
	Noisy *D*2	GNG-M	4.0±0.5	0.647±0.188
		RGNG	3.2±0.4	0.860±0.080
		p-value	0.001	0.004

The data set is normalized between [−1,1].

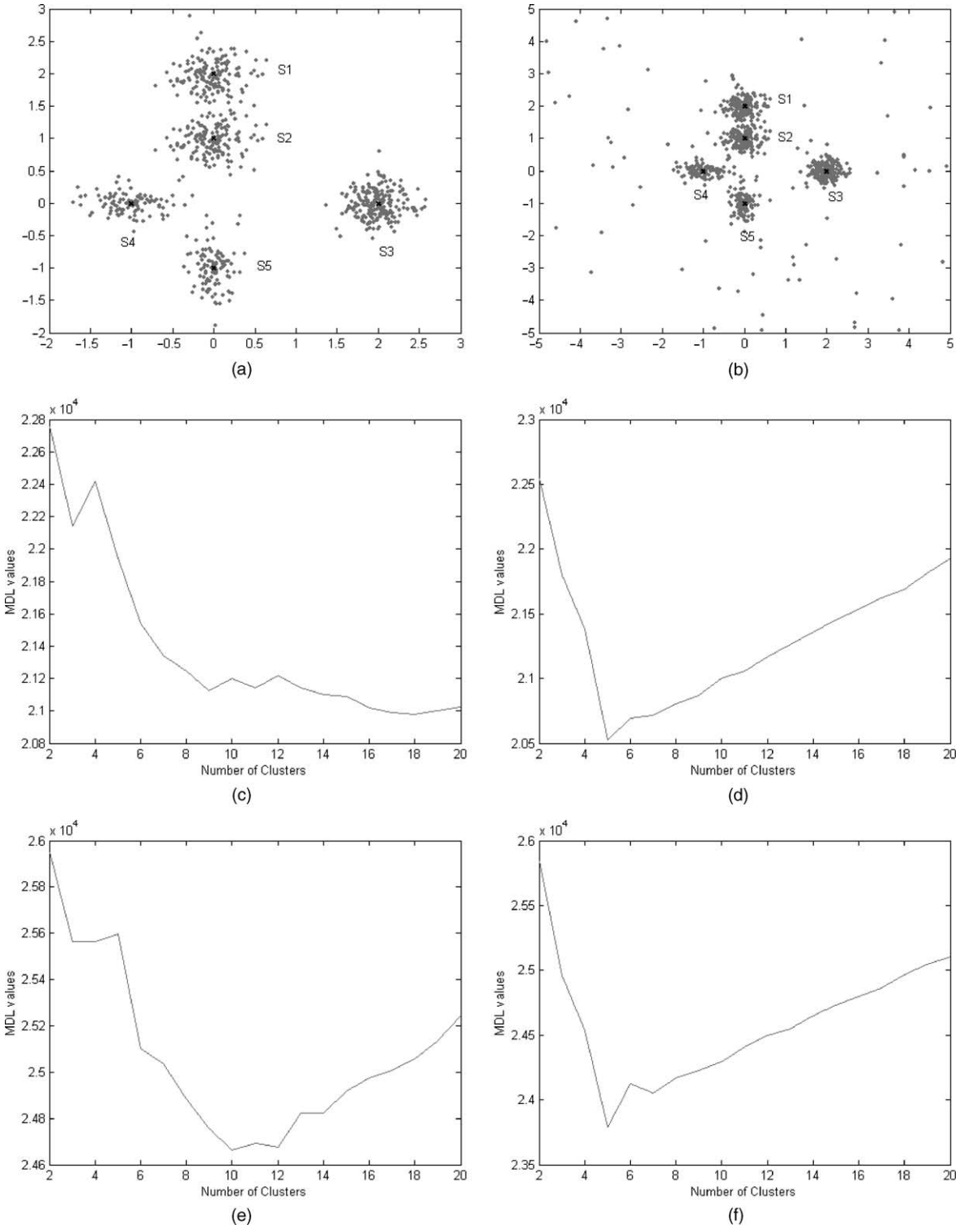

Fig. 1. (a) Plot of clean data set D_1. (b) Plot of noisy data set D_1. (c,d) Plot of the MDL values versus the number of prototypes during growth process of GNG-M and RGNG algorithm on clean D_1. (e,f) Plot of the MDL values versus the number of prototypes during growth process of GNG-M and RGNG algorithm on noisy D_1.

inferior performance as it fails to find the correct cluster number 25 on both clean and noisy D_2 and the PQ measure possesses much lower values than those yielded by the RGNG algorithm. In addition, when the number of prototypes is the same as the actual number of clusters during the growing process, the average number of natural clusters No_Cluts detected by the GNG-M over the 10 runs significantly deviates from the actual value of 25 and the

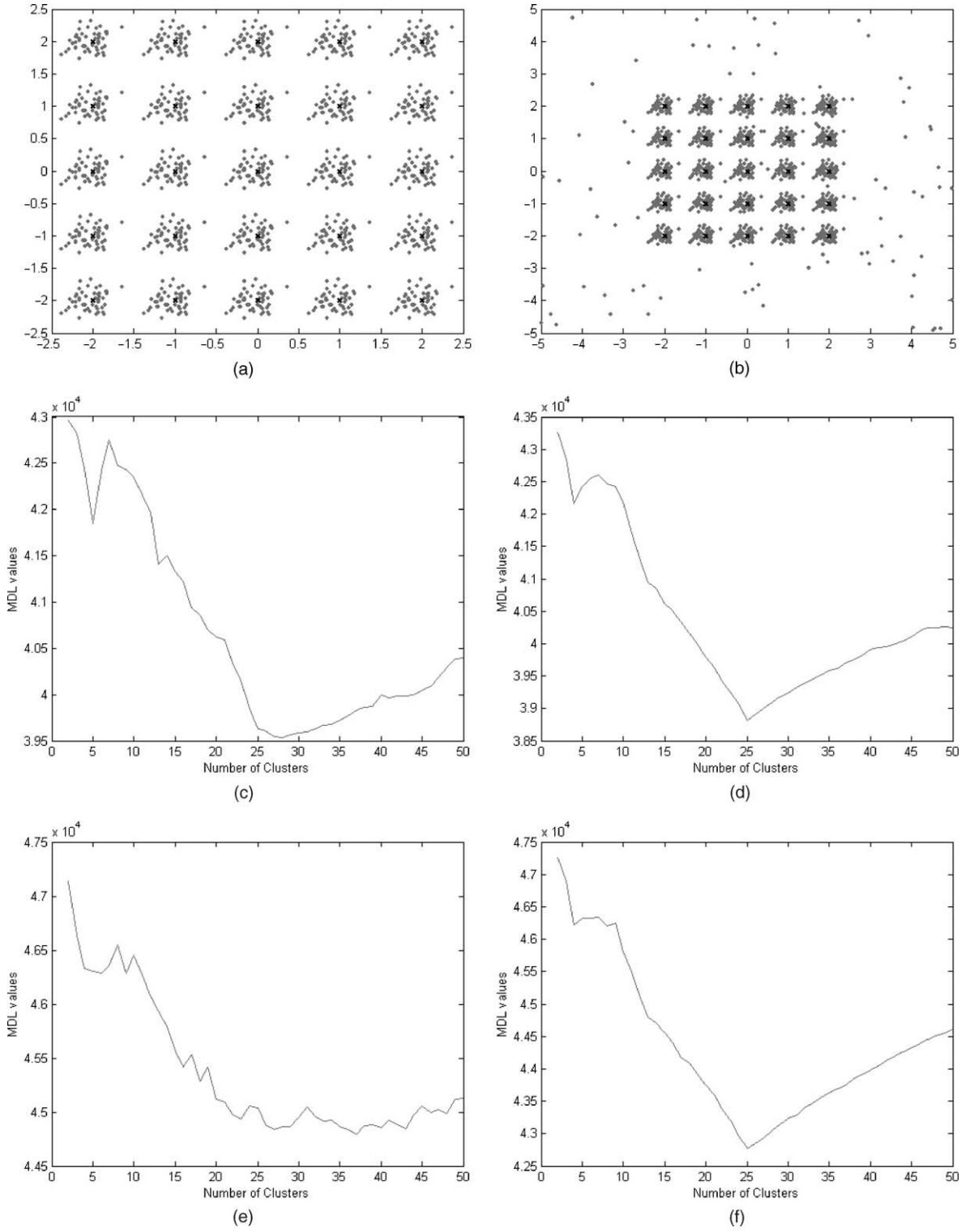

Fig. 2. (a) Plot of clean multimodal data set D_2. (b) Plot of noisy multimodal data set D_2. (c,d) Plot of the MDL values versus the number of prototypes during growth process of GNG-M and RGNG algorithm on clean D_2. (e,f) Plot of the MDL values versus the number of prototypes during growth process of GNG-M and RGNG algorithm on noisy D_2.

corresponding average MSE value is much higher. In contrast to the GNG-M algorithm, the RGNG possesses higher accuracy and robustness in both clean and noisy data sets, as evidenced by the small p-values shown in Tables 3 and 4.

4.2. UCI data

We tested our algorithm on one UCI benchmark data set 'wine' (Blake, Keogh, & Metz, 1998) to demonstrate the performance of our algorithm on a real data clustering

task In order to highlight our algorithm's promising robustness, we also apply the two clustering algorithms to a noisy version of wine data set which is generated by adding two outliers, far away from the original sample points, to contaminate the original data set. Since the purpose of clustering task is to group the data set into several clusters without any guided information, we first discarded the class label information attached in the data set and then applied our RGNG and the GNG-M algorithms on the unlabeled data set. In real data sets, usually the number of natural clusters is not equal to the number of classes due to the possibility that one class may have many sub-clusters. Therefore, there is little knowledge about the actual number of clusters. Here we employ the *PQ* value to evaluate the algorithm's clustering performance. By comparing the *PQ* values representing the partitioning quality with the detected optimal number of clusters by GNG-M and RGNG algorithms, the RGNG algorithm shows better partitioning results with its obtained optimal cluster number in both noisy and clean conditions.

The clustering results in Table 5 indicate that the two clustering algorithms detect different number of clusters. With the respective number of clusters obtained, our RGNG algorithm can achieve higher *PQ* value than the GNG-M algorithm, which means that the RGNG finds more representative clusters.

5. Conclusions and future works

In this paper, we have presented a novel Robust Growing Neural Gas algorithm, called RGNG algorithm. By incorporating robust strategies, such as outlier resistant scheme, adaptive learning rates modulation and cluster repulsion method into the Growing Neural Gas framework, the RGNG is made insensitive to the initializations, input sequence ordering and the presence of outliers. Furthermore, the optimal number of clusters corresponding to the extreme MDL value over a certain number of growth stages can be detected. Experimental results have shown improved robustness of the RGNG over the GNG incorporating MDL (i.e. GNG-M) method on both artificial and UCI data sets in static data clustering tasks.

Due to utilizing the Euclidean metric, now our algorithm is able to discover hyper-spherical and hyper-ellipsoidal clusters with small difference in the variance in each dimension. However, the clusters of various shapes can be detected by employing other distance measures (Frigui & Krishnapuram, 1999) in the objective function to be optimized. Hence, incorporating different distance metric into our clustering algorithm can be one of our future research directions. Another research direction is to kernelize the RGNG algorithm by using Mercer kernel functions (Girolami, 2002). By mapping data points into the higher dimensional feature space, the nonlinear cluster boundaries commonly existing in real data sets can be transformed into linear ones, therefore the RGNG algorithm may perform better in the feature space.

The robustness strategies employed in our RGNG algorithm can also be generalized to many GNG based learning frameworks (Cao & Suganthan, 2003; Carlevarino, Martinotti, Metta, & Sandini, 2000). We expect that, in noisy environments, the performance of the original GNG based methods may not deteriorate much if the robust properties presented in this paper are incorporated. Since our algorithm can deal well with multimodal data sets, we also plan to apply our algorithm to real multimodal data sets, such as segmentation of MRI images, which usually possess multimodality.

Acknowledgements

We would like to express our deepest gratitude to all anonymous reviewers for their valuable comments

References

Ahalt, S. C., Krishnamurty, A. K., Chen, P., & Melton, D. E. (1990). Competitive learning algorithms for vector quantization. *Neural Networks*, *3*(3), 277–291.

Atukorale, A. S., Downs, T., & Suganthan, P. N. (2003). Boosting the HONG network. *Neurocomputing*, *51*, 75–86.

Berkhin, P. (2002). *Survey of clustering data mining techniques* Accrue Software, http://www.accrue.com/products/rp_cluster_review.pdf.

Bezdek, J. C., Keller, J. M., Krishnapuram, R., & Pal, N. R. (1999). *Fuzzy models and algorithms for pattern recognition and image processing*. Norwell, MA: Kluwer.

Bhatia, S. K., & Deogun, J. S. (1998). Conceptual clustering in information retrieval. *IEEE Transactions on System, Man, Cybernetics, Part B*, *28*(3), 427–436.

Bischof, H., Leonardis, A., & Selb, A. (1999). MDL principle for robust vector quantization. *Pattern Analysis and Application*, *2*(1), 59–72.

Blake, C., Keogh, E., & Metz, C. J. (1998). *UCI repository of machine learning databases*. Department of Information and Computer Science, University of California at Irvine, CA, http://www.ics.uci.edu/(mlearn/MLRepository.html.

Cao, X., & Sugantahn, P. N. (2003). Video shot motion characterization based on hierarchical overlapped growing neural gas networks. *Multimedia Systems*, *9*(4), 378–385.

Carlevarino, A., Martinotti, R., Metta, G., & Sandini, G. (2000). Incremental growing neural network and its application to robot control. *Proceeding of the 2000 international joint conference on neural networks (IJCNN00), 5* (pp. 323–328), Como, Italy.

Chintalapudi, K. K., & Kam, M. (1998). The credibilistic fuzzy *c*-means algorithm. *Proceeding of 1998 IEEE international conference on systems, man and cybernetics, San Diego, CA* , 2034–2040.

Clark, M. C., Hall, L. O., Goldgof, D. B., Clarke, L. P., Velthuizen, R. P., & Silbiger, M. S. (1994). MRI segmentation using fuzzy clustering techniques. *IEEE Engineering in Medicine and Biology Magazine*, *13*, 730–742.

Dave, R. N., & Krishnapuram, R. (1997). Robust clustering methods: a unified view. *IEEE Transactions on Fuzzy Systems*, *5*, 270–293.

Duda, R., & Hart, P. (1973). *Pattern recognition and scene analysis*. New York: Wiley/Interscience.

Frigui, H., & Krishnapuram, R. (1997). Clustering by competitive agglomeration. *Pattern Recognition*, *30*(7), 1109–1119.

Frigui, H., & Krishnapuram, R. (1999). A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis Machine Intelligence*, *21*, 450–465.

Fritzke, B. (1994). Growing cells structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*(9), 1441–1460.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems* (pp. 625–632). Cambridge, MA: MIT Press.

Fritzke, B. (1997). *Some competitive learning methods (draft)*. Technique report. Institute for Neural Computation, Ruhr-University, Bochum, http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/.

Girolami, M. (2002). Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, *13*, 780–784.

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

Haykin, S. (1998). *Neural networks: a comprehensive foundation* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Haykin, S. (2001). *Adaptive filter theory* (4th ed.). Englewood Cliffs, NJ: Prentice-Hall.

Jain, I. K., Murty, N. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 264–323, http://www.acm.org/pubs/citations/journals/surveys/1999-31-3/p264-jain/.

Kato, N., & Nemoto, Y. (1996). Large scale handwritten character recognition system using subspace methods. *Proceeding of 1996 IEEE international conference on systems, man and cybernetics, Beijing, China*, *1*, 432–437.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin: Springer.

Martinetz, T. M. (1993). Competitive Hebbian learning rule forms perfectly topology preserving maps. *Proceeding of 1993 international conference on artificial neural networks (ICANN93)* (pp. 427–434), Amsterdam, The Netherlands.

Martinetz, T. M., Berkovich, S. G., & Schulten, K. J. (1993). Neural gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*, 558–569.

Martinetz, T. M., & Schulten, K. (1994). Topology representing networks. *Neural Networks*, *3*, 507–522.

Pal, N. R., Pal, K., & Bezdek, J. C. (1997). A mixed *c*-means clustering model. *Proceeding of 1997 IEEE international conference on fuzzy system*, Spain.

Pelleg, D., & Moore, A. (2000). X-means: Extending *k*-means with efficient estimation of the number of clusters. *Proceeding of the 17th international conference on machine learning* (pp. 727–734), San Francisco, USA.

Qin, A. K., & Suganthan, P. N. (2004). A robust neural gas algorithm for clustering analysis. *Proceeding of 2004 international conference on intelligent sensing and information processing (ICISIP2004)* (pp. 342–347), Chennai, India.

Ray, S., & Turi, R. H. (1999). 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)* (pp. 137–143), Calcutta, India.

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

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry World Scientific: series in computer science, 15*.

Tenmoto, H., Kudo, M., & Shimbo, M. (1998). MDL-based selection of the number of components in mixture models for pattern classification. In *Advance* in *Pattern Recognition*, edited by A. Amin, D. Dori, P. Pudil & H. Freeman, *Lecture Notes in Computer Science, Springer*, no. 1451, pp. 831–836.

Timm, H., Borgelt, C., Döring, C., & Kruse, R. (2001). *Fuzzy cluster analysis with cluster repulsion Proceeding of European symposium on intelligent technologies, hybrid systems and their implementation on smart adaptive systems, Germany*.

Winter, M., Metta, G., & Sandini, G. (2000). Neural-gas for function approximation: a heuristic for minimizing the local estimation error. *Proceeding of 2000 international joint conference on neural network (IJCNN00), 4* (pp. 535–538), Italy.

Wu, K.-L., & Yang, M.-S. (2002). Alternative *c*-means clustering algorithms. *Pattern Recognition*, *35*, 2267–2278.

Yang, T.-N., Wang, S.-D., & Yen, S.-J. (2002). Fuzzy algorithms for robust clustering. *Proceeding of international computer symposium, Houlian, Taiwan*.

Yu, H. (1998). *Automatically determining number of clusters*. Information retrieval (CMU CS11-741) final report.

Zemel, R. S. (1994). *A minimum description length framework for unsupervised learning*. PhD thesis, University of Toronto.

A.K. Qin received his B.B. Degree from Department of Automatic Control Engineering of Southeast University, Nanjing, P.R. China in 2001. From 2003, he studied towards his Ph.D. degree in the School of Electrical and Electronic Engineering, Nanyang Technological University. His research interests include machine learning, pattern recognition, neural network, evolutionary algorithms and bioinformatics. He is a member of Pattern Recognition and Machine Intelligence Association, Singapore.

P.N. Suganthan received the BA degree, Postgraduate Certificate and MA degree in Electrical and Information Engineering from the University of Cambridge, UK in 1990, 1992 and 1994, respectively. He obtained his PhD degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He was a predoctoral Research Assistant in the Department of Electrical Engineering, University of Sydney in 1995–1996 and a lecturer in the Department of Computer Science and Electrical Engineering, University of Queensland in 1996–1999. Between 1999 and 2003, he was an Assistant Professor in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore where he is now an Associate Professor. He is an associate editor of the Pattern Recognition Journal. His research interests include evolutionary computation, applications of evolutionary computation, neural networks, pattern recognition and bioinformatics. He is a senior member of the IEEE and an associate member of the IEE.