Growing Neural Gas for Temporal Clustering

Isaac J. Sledge and James M. Keller
Electrical and Computer Engineering Department, University of Missouri
ijs3h6@missouri.edu, kellerj@missouri.edu

Abstract

Conventional clustering techniques provide a static snapshot of each vector's commitment to every group. With additive datasets, however, existing methods may not be sufficient for adapting to the presence of new clusters or even the merging of existing data-dense regions. To overcome this deficit, we explore the use of growing neural gas for temporal clustering and provide evidence that this new algorithm is capable of detecting cluster structures that incrementally emerge.

1. Introduction

Humans have the distinct advantage of efficiently differentiating between unlike objects while simultaneously categorizing similar ones. Computers, too, can be endowed with similar capabilities in the form of unsupervised clustering methods. During exploratory data analysis, a set of objects, $\mathbf{X} = \{\vec{x}_1, ..., \vec{x}_n\}^T \subset \mathbb{R}^s$, is organized into c self-similar subsets based upon an underlying similarity measure. Though there are a myriad of different measures, the most developed of these involves optimizing the relationship between the spatial location of a prototype set, $\mathbf{V} = \{\vec{v}_1, ..., \vec{v}_c\} \subset \mathbb{R}^s$, and the set of objects, such that each $\vec{v}_j$ captures a compact representation of a clusters structure. Due to its prevalence, many well-known methods fall into this model, with some that extend the prototypical set to include linear varieties and hyperspherical shells [1]. However, many of these techniques have two critical shortcomings: they are reliant on a specified cluster count and the produced c-partition is localized in time.

Given an unlabeled, temporally growing dataset, $\mathbf{X}^t = \{\vec{x}_1^t, ...\} \subset \mathbb{R}^s$, are there any mechanisms that can automatically determine the number of clusters while succinctly evolving a membership partition, $U(\mathbf{X}^t) = [u_{j,i}]_{c^t \times n^t}$, $j \in \mathbb{R}_{c^t}$, $i \in \mathbb{R}_{n^t}$, as new data is introduced? Without *a* priori knowledge of the data distribution, various tendency assessment algorithms, such as cluster validity metrics [1] or any the recently emerged batch of visually-based measures [2], can be executed to return an estimate, c_{est}, of the number of coherent groups. In addition, at least one deterministic approach, the unsupervised fuzzy partition optimum number of classes, is capable of learning the value of c_{est}, in an online fashion, by coalescing a fuzzy maximum likelihood estimate with the fuzzy c-means algorithm [1]. But, none of these methods are sufficient for temporal applications, as the results are static and cannot adapt to new data without re-executing the algorithm. To satisfy these dynamic needs, concepts from the realms of competitive Hebbian learning, adaptive resonance theory (ART) [3], and computational geometry can be married to form a new algorithm capable of exploring hyperspace and autonomously segmenting groupings of objects. We, henceforth, refer to this neurally-inspired scheme as growing neural gas clustering (GNGC), since the spirit of the approach stems from growing neural gas [4].

To understand how these neuronal forces can be harnessed to automatically cluster incrementally added data, a brief canvass of growing neural gas (GNG) is presented in Sec. II. This leads into the formulation of a GNG framework for temporal exploratory data analysis of hyperspherical, hypershellular and hyperplanar structures. To empirically verify that the new method functions appropriately, Sec. III introduces the list of conducted experiments, discusses the results, and summarizes the findings. This segues into Sec. IV, which concludes the construction of this new tool.

2. Neuronal Clustering

As an excellent tool for topological learning, growing neural gas parallels earlier, biologically influenced, methods for signal approximation. The modus operandi of this approximation, or vector quantization (VQ), is to encode each manifold, $M \subseteq \mathbb{R}^s$, of signals using a set, $\mathbf{W} = \{\vec{w}_1, ..., \vec{w}_m\} \subset \mathbb{R}^s$, of reference vectors. For GNG, this is facilitated by adapting the best-matching $\vec{w}_k$, and its connected neighbors on a dynamic lattice structure, for each presented input stimulus. Since there are no explicit constraints on the neural nets topological arrangement, new connections can be forged, between arbitrary, non-connected $\vec{w}_k$'s, based upon the induced magnitude response of each $\vec{w}_k$'s receptive fields. In addition, obsolete connections are allowed to die out, due to an 'aging' factor. Combined with the lack of any parameter decay and the ability to iteratively add new neuronal reference vectors to the network, these algorithmic facets enable GNG to locate and approximate a gamut of data structures.

While primarily founded as a method for VQ, the applications of GNG are not bound solely to this domain and can instead flow over into analogous areas. Consequently, the algorithm is particularly attractive for temporal clustering, due to its incremental style of learning, which can be extended to handle additive data and adapt $U(\mathbf{X}^t)$ accordingly. Many additional benefits can also be unearthed and reaped from this approach, such as the capability to determine c^t, in an online fashion, without the need for a minimal representation of the class distribution at initialization or the concurrent execution of any cluster validity techniques. This contrasts with earlier attempts to address the clustering of temporal inputs in the form of data streams [5-6] and speech signals [7-8]. Furthermore, GNG can be tweaked to function even when the entire dataset grows beyond the limits of physical memory, as unnecessary data vectors can be pruned and partially accounted for via VQ. Embodying these aspects, among others, into GNGC, requires only a few novel extensions of the base GNG algorithm, which we elucidate in Alg. 1.

Similar to the standard GNG algorithm in [4], GNGC begins by randomly initializing two reference vectors, $\vec{w}_1$, $\vec{w}_2 \subset \mathbb{R}^s$, which are connected together via an undirected graph edge with an age of zero. After generating the references, a random datum, $\vec{x}_i^t$, is selected from the temporally added dataset, $\mathbf{X}^t$, and compared to the

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

Algorithm 1: Growing Neural Gas Clustering

Input: $\mathbf{X}^t \subset \mathbb{R}^s$: temporally growing dataset; $w_{\max} \in \mathbb{N}^*$: max. number of reference vectors; $a_{\max} \in \mathbb{N}^*$: max. arc age; ϵ_b, $\epsilon_n \in \mathbb{R}^+$: distance scalars; α, $\beta \in \mathbb{R}^+$: error scalars; $\lambda \in \mathbb{N}^*$: iteration counter; $\sigma \in \mathbb{N}^*$: reference increment counter, $\gamma \in \mathbb{R}^+$: distance threshold

Data: $U(\mathbf{X}^t)$: partition matrix; $\mathbf{W}$: a list of references; $w_{\text{cur}} \in \mathbb{N}^*$: current number of reference vectors; w_{con}: a list of reference connections, connection ages and connection errors

Initialize $\mathbf{W}$ with two references, at random locations bounded by $\mathbf{X}^t$, with an error of zero; Create an arc between the two references, in w_{con}, and set its age to zero; Set $w_{\text{cur}} = 2$

while *new data available* **or** *waiting for new data* **do**

- **if** σ *new data vectors have been appended to* $\mathbf{X}^t$ **then**
  - Increment $w_{\max}$
- Randomly select an input data vector, $\vec{x}_i^t \in \mathbf{X}^t$
- Find the two closest reference vectors, $\vec{w}_{k_1}, \vec{w}_{k_2} \in \mathbf{W}$, w.r.t. $\vec{x}_i^t$
- **if** $||\vec{x}_i^t - \vec{w}_{k_1}||^2 \geq \gamma$ **and** $w_{\text{cur}} < w_{\max}$ **then**
  - Spawn new reference, $\vec{w}_r$, at $\vec{x}_i^t$
  - Create arc, with age zero, between $\vec{w}_r$ and $\vec{w}_{k_1}$
  - Initialize error of $\vec{w}_r$, in w_{con}, with the new error of $\vec{w}_{k_1}$
- **else**
  - Increment the age, in w_{con}, of all arcs emanating from $\vec{w}_{k_1}$
  - Increase the error, in w_{con}, of $\vec{w}_{k_1}$ by $||\vec{w}_{k_1} - \vec{x}_i^t||^2$
  - Move $\vec{w}_{k_1}$ and its topological neighbors, $\vec{w}_{k_n}$, towards $\vec{x}_i^t$: $\Delta\vec{w}_{k_1} = \epsilon_b\left(\vec{x}_i^t - \vec{w}_{k_1}\right)$, $\Delta\vec{w}_{k_n} = \epsilon_n\left(\vec{x}_i^t - \vec{w}_{k_1}\right)$
  - **if** $\vec{w}_{k_1}$ *and* $\vec{w}_{k_2}$ *are connected by an arc* **then**
    - Set age of arc, in w_{con}, between $\vec{w}_{k_1}$ and $\vec{w}_{k_2}$ to 0
  - **else**
    - Create arc between $\vec{w}_{k_1}$ and $\vec{w}_{k_2}$, in w_{con}, with age 0
- Remove any arcs, in w_{con}, with age greater than $a_{\max}$
- **if** *exists, in* w_{con}*, any* $\vec{w}_k$ *with no emanating arcs* **then**
  - Remove $\vec{w}_k$ from $\mathbf{W}$ and its entry in w_{con}; Decrement w_{cur}
- **if** λ *iterations have executed* **and** $w_{\text{cur}} < w_{\max}$ **then**
  - Determine the reference vector, $\vec{w}_q$, with the max. error
  - Increment w_{cur}; Insert a new reference vector, $\vec{w}_r$, into $\mathbf{W}$, halfway between $\vec{w}_q$ and its neighbor, $\vec{w}_f$, with the largest error: $\vec{w}_r = 0.5(\vec{w}_q + \vec{w}_f)$
  - Insert arcs, in w_{con}, connecting $\vec{w}_q$ to $\vec{w}_r$ and $\vec{w}_r$ to $\vec{w}_f$
  - Scale the errors of $\vec{w}_q$ and $\vec{w}_f$, in w_{con}, by α
  - Initialize error of $\vec{w}_r$, in w_{con}, with the new error of $\vec{w}_q$
- Scale the errors of all references, in w_{con}, by β
- Compute non-Hamiltonian path of all arcs, in w_{con}, via a depth-first search; Increment the number of clusters, c, for each path
- **for** $j = 1$ **to** c **do**
  - **if** *number of* $\vec{w}_k$*'s in path* j *less than* $s + 1$ **then**
    - Find $\vec{v}_j$ as mean of $\vec{w}_k$'s in path j; Produce a fuzzy (1) or possibilistic (2) $U(\mathbf{X}^t)$ with $d_{j,i}$ in (5)
  - **else**
    - Calculate the convex hull of the $\vec{w}_k$'s [9]
    - Determine the shape of the j^{th} cluster; Estimate the mean, $\vec{v}_j$; Produce a fuzzy (1) or possibilistic (2) $U(\mathbf{X}^t)$ with $d_{j,i}$ from (3)-(5)
- Increment iteration counter

current set of reference vectors in $\mathbf{W}$. The two nearest references, $\vec{w}_{k_1}, \vec{w}_{k_2}$, with respect to $\vec{x}_i^t$, are then found using a specified distance metric. To adapt the topology of the dynamic reference graph, so that it better matches M, a check is performed to ascertain if the distance between $\vec{w}_{k_1}$ and $\vec{x}_i^t$ is greater than some user-specified threshold, γ. If the conditional evaluates to true, then ART is invoked and a new reference, $\vec{w}_r$, is spawned at $\vec{x}_i^t$ and connected to $\vec{w}_{k_1}$ via an undirected graph edge with an age of zero. This approach not only helps to increase the stability of previously learned clusters but also promotes plasticity for exploring forming structures. However, if the distance is less than γ, then the regular GNG reference update is performed.

While spatial reference adaptation is important for incrementally learning M, it is also critical that new reference nodes are successively introduced and connected to the existing network. Provided that the number of iterations is an integer multiple of some user-selected λ, and that the current number of $\vec{w}_k$'s, w_{cur}, is less than the maximum number of references, $w_{\max}$, a new node is inserted. This is facilitated by finding the $\vec{w}_k$ with the highest amount of accumulated error, $\vec{w}_q$, and creating a new reference, $\vec{w}_r$, halfway between it and its connected neighbor with the highest accumulated error. To aid in learning new topologies, the maximum number of references is allowed to grow as a function of $\mathbf{X}^t$.

The largest difference, however, in comparison to the GNG algorithm, is from the latter part of Alg. 1, whereby the cluster type recognition and partition generation are fully automated, provided the number of $\vec{w}_k$'s in path j is greater than the dataset dimensionality. A depth-first search (DFS) is performed, up front, to isolate non-connected neural topologies. This subsequently triggers the convex hull computation, of the DFS path, which yields a convex polytope, $\mathcal{P}_j \in \mathbb{R}^s$. At this point, if there is a priori knowledge concerning the structure types, a decision can be made by the user as to whether clustering should be carried out for hyperspheres, hypershells or hyperplanes. If this knowledge is not available, then the eigenvalues of the covariance matrix for $\mathcal{P}_j$ are found. Using this piece of information, it is easy to discern a measure of "flatness" for $\mathcal{P}_j$, as hyperlinear, and approximately hyperlinear, structures will have at least one eigenvalue that is centered closely about zero. A simple threshold check can be carried out, and if any eigenvalue is less than a small, user selected threshold, then $\mathcal{P}_j$ is assumed to be hyperlinear; otherwise, it is assumed to span all dimensions in s and thus should be considered a non-linear manifold. Density testing can then be performed to distinguish between shellular and spherical groups. This check, for hyperspherical clusters, depends on the veracity of: $\frac{n_{\mathcal{P}}}{a_{\mathcal{P}}} > \psi$, where $n_{\mathcal{P}}$ is the number of data vectors inside $\mathcal{P}_j$, which is found by using an extension of the Jordan-Brouwer theorem [9], $a_{\mathcal{P}}$ the surface area of $\mathcal{P}_j$, and ψ a user-specified threshold. Similarly, if $\frac{n_{\mathcal{P}}}{a_{\mathcal{P}}} \leq \psi$, the cluster is assumed to be hypershellular.

After the shape of the cluster has been determined, and the cluster prototypes found, the memberships of the points are calculated. For the hyperspherical case, the mean of the points in the polytope is calculated, and a fuzzy or possibilistic partition is computed via:

$$u_{j,i} = \left(\sum_{q=1}^{c} \left(d_{j,i}/d_{q,j}\right)^{\frac{2}{m-1}}\right)^{-1}, \forall j, i$$
$$u_{j,i} \in [0,1]\ \forall j, i, \textstyle\sum_{j=1}^{c} u_{j,i} = 1\ \forall i \tag{1}$$

$$u_{j,i} = \left(1 + \left(d_{j,i}^2/\eta_j\right)^{\frac{1}{m-1}}\right)^{-1}, \forall j, i$$
$$u_{j,i} \in [0,1]\ \forall j, i, \textstyle\sum_{i=1}^{n} u_{j,i} \leq n\ \forall i, \max_i u_{j,i} > 0 \tag{2}$$

where $d_{j,i}$ is the distance from $\vec{x}_i^t$ to the j^{th} prototype, $m \geq 1$ a degree of fuzzification and $\eta_j \geq 0$ the distance at which the membership of a point becomes 0.5 [1]. For hypershellular clusters, the cluster centroid, $\vec{v}_j$, is estimated as the barycenter of $\mathcal{P}_j$ and $U(\mathbf{X}^t)$ is formed using either (1) or (2), with radial distance measure:

$$d_{j,i}^2 = \frac{\left(||\vec{x}_i^t - \vec{v}_j||_{A_j} - 1\right)^2 ||\vec{x}_i^t - \vec{v}_j||^2}{||\vec{x}_i^t - \vec{v}_j||_{A_j}^2} \tag{3}$$

where A_j is a positive, definite, symmetric matrix accounting for the eccentricity and orientation of the hypershell [3]. The value of A_j, for this equation, is approximated by finding the Löwner hyperellipsoid, of the convex polytope $\mathcal{P}_j$, using the Khachiyan algorithm [10]. Likewise, points in hyperplanar structures are assigned membership using (1)-(2), where:

$$d_{j,i}^2 = ||\vec{x}_i^t - \vec{v}_j||_A^2 - \sum_{k=1}^{s} \langle \vec{x}_i^t - \vec{v}_j, \vec{b}_{j,k} \rangle_A^2 \quad (4)$$

is the orthogonal distance from x_i^t to the j^{th} variety, in $\mathbb{R}^s$, and A is an arbitrarily defined positive, definite, symmetric weight matrix [1]. In (4), $\vec{v}_j$ is approximated as the barycenter of $\mathcal{P}_j$, while $\vec{b}_{j,k}$ is estimated by finding the eigenvectors of the neuronal references that model the linear variety. Finally, for point-cloud (hyperspherical) clusters, $d_{j,i}$ is found as:

$$d_{j,i}^2 = ||\vec{x}_i^t - \vec{v}_j||^2 \quad (5)$$

where $\vec{v}_j$ is either the mean of the data enclosed by $\mathcal{P}_j$, if the convex hull can be computed, or the mean of the reference vectors in the j^{th} path.

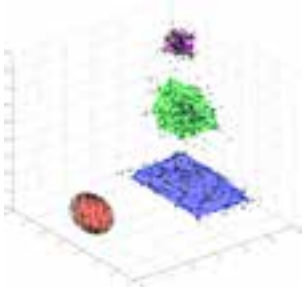
2.1. High Dimensional Visualization

When working with low dimensional data, say $\mathbf{X} \subset \mathbb{R}^3$, it is easy to display intermittent GNGC results, such as those in Fig. 1(a). However, if the dimensionality of the dataset grows beyond $\mathbb{R}^3$, capturing the same spatial information and visualizing the learned distributions is problematic. To rectify this issue, several conventions were borrowed from the visual assessment of [cluster] tendency (VAT) algorithm [13]. In VAT, a matrix, $\mathbf{R} = [r_{j,i}]_{n \times n}$, of normalized, pair-wise dissimilarity values, are ordered using a modified variant of Prim's algorithm for finding a minimal spanning tree (MST). After the sorting process, if the matrix is displayed as an intensity image then cluster structure is indicated by the presence of dark blocks along the main diagonal.

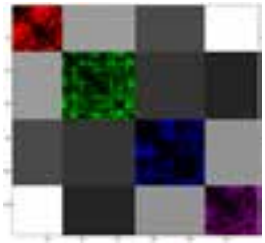
By modifying the VAT approach, to instead use information obtained from GNGC, we can create images like those in Fig. 1(b) and 1(c). These plots, which we call neuronal dissimilarity images (NerDI) [12], are generated by first computing the normalized, pairwise dissimilarity of the neurons in each isolated graph. Utilizing Prim's modified MST process, the dissimilarities are rearranged, for each path, which aids in looking for dense, intra-cluster distributions of neurons. Each of these sub-matrices are then colored and placed along the main diagonal of the NerDI. Inter-cluster spatial relationships are also incorporated in the image by finding the minimal distance between the neurons in each path; these values are then normalized so that white denotes the largest distance between two clusters, in $\mathbb{R}^s$, and black the smallest. Finally, volume or clusterness [13] information can be added, as a third dimension, to provide additional insight about the approximated manifolds.

3. Experiments and Results

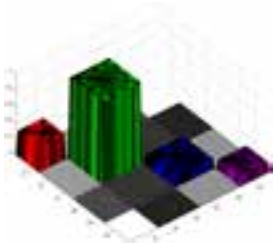
To test the effectiveness of growing neural gas clustering, and its ability to temporally cluster a variety of cluster types, a complex synthetic dataset was generated. Applications with real-world, high dimensional data can be found in Sledge et. al [12] and Sledge, Keller and Alexander [14].



(a) Synthetic data with learned $\vec{w}_k$'s and colored convex hulls



(b) Corresponding neuronal dissimilarity image for (a)



(c) NerDI with normalized polytope volume information

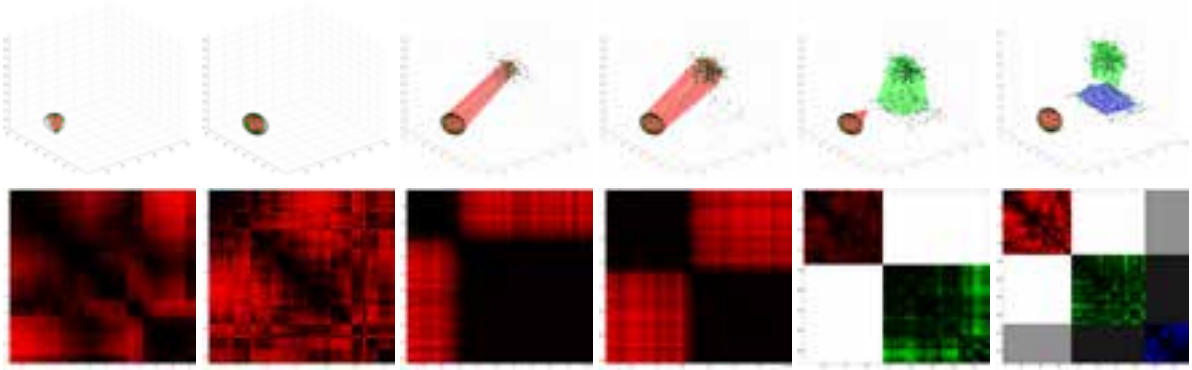
Figure 1: GNGC results showing four clusters, along the main diagonal of (b) and (c), and the spatial similarities between the learned distributions, using gray-scale values. In (b) and (c), the purple cluster is near the green cluster, somewhat close to the blue planar cluster, and far from the red spherical cluster. In (c), the purple cluster has the smallest convex polytope volume.

3.1. Synthetic Dataset Experiment

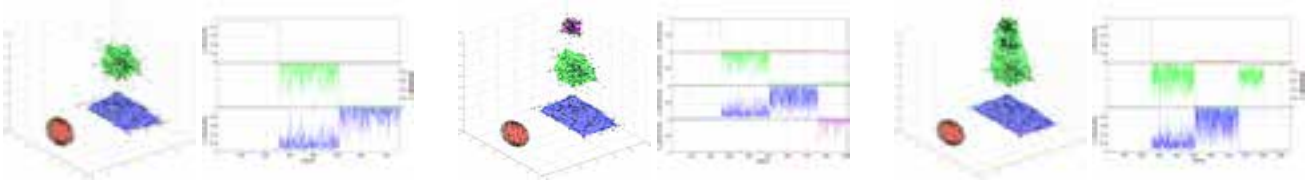
In the preceding section, a claim was made that GNGC could not only detect incrementally emerging object-data clusters, but also determine the structures overall shape. For validating this assertion, a 1050-point dataset, $\mathbf{X}_1^t \subset \mathbb{R}^3$, was created to contain all three cluster structure types. Over a span of 5000 iterations, 75% of the data was incrementally fed to the GNGC algorithm, with the remaining 25% later introduced to illustrate cluster merging. To drive the clustering process, the following parameters are used: $w_{\max} = 30$, $a_{\max} = 15$, $\epsilon_b = 10^{-1}$, $\epsilon_n = 10^{-4}$, $\alpha = 10^{-2}$, $\beta = 10^{-4}$, $\lambda = 50$, $\sigma = 10$, $\gamma = 2$, $m = 1.5$, and η_j as twice the average distance from the cluster center to each vertex of $\mathcal{P}_j$.

Once a small fraction of the 756 data points were introduced to the system, the neuronal reference vectors began to adapt to the forming distributions, as shown in Fig. 2(a). After a total of 7500 iterations, the average neuronal error planed, denoting that the structures were sufficiently approximated by the $\vec{w}_k$'s, and a confident measure of the memberships, $U(\mathbf{X}_1^t)$, was computed. Comparing the color-coded values of $U(\mathbf{X}_1^t)$, shown in Fig. 2(b), with its corresponding data distribution plot, it is apparent that the GNGC algorithm noted the presence of three spatially dense regions. Additional evidence was also provided, by the algorithm, that the shellular (red), spherical (green), and planar (blue) regions in Fig. 2(b) were properly classified via the autonomous shape detection scheme.

While GNGC appears to be working, one disconcerting phenomenon was the moderately high memberships, in the planar cluster, for the points in the spherical distribution. Since the orthogonal distance metric should bias against non-collinear structures, the overall membership for these points should be significantly dampened. But, upon delving into the problem, it was determined that the combination of a high value for η_j, the nature of the possibilistic function, and the close proximity of the two clusters produced the



(a) Temporal plots, and the corresponding NerDIs, of the GNGC clustering results for the first part of the three shapes dataset. As more data is added to $\mathbf{X}_1^t$, the $\vec{w}_k$'s, shown using green spheres, adapt their location and connectivity to better model the data. Note that the third and fourth NerDI plots highlight a high dissimilarity between the $\vec{w}_k$'s in the red colored convex hull.



(b) Partial data plot with convex hulls (c) Emergence of a new cluster (shown in purple) (d) Amalgamation of two clusters (shown in green)

Figure 2: Growing neural gas clustering results and cluster memberships for the synthetic shapes dataset.

effect. Had fuzzy memberships instead been found and displayed, these erratic fluctuations would have been far less pronounced.

Knowing that all of the algorithms components were functioning properly, a set of 150 new vectors was injected, randomly, into $\mathbf{X}_1^t$. This resulted in the formation of a new spherical class, which is denoted in Fig. 2(c) using a magenta color. Despite the spatial propinquity of the green and magenta regions, the GNGC algorithm noted the increase in the cluster count and adapted $U(\mathbf{X}_1^t)$ accordingly. At this point, the remaining 120 object data were iteratively interposed, between the green and magenta clusters, to bridge the two regions. GNGC subsequently updated both the dynamic network, to reflect the topological change illustrated in Fig. 2(d), and $U(\mathbf{X}_1^t)$, as the overall cluster count decreased. As with the previous results, those in Fig. 2(c)-(d) met our expectation.

4. Conclusions

In this paper, we introduced a method for temporal clustering of additive datasets using a modified GNG approach. Unlike previous research, GNGC requires no *a priori* information about the data, such as an estimate of the number of clusters or cluster shape. Instead, our autonomous implementation will discern this pertinent information during execution.

Although the results, thus far, have been favorable, there are several extensions that can be considered to improve the usability of our algorithm. Foremost, temporal information can be fused with the approach to provide a different style of learning. As well, the automatic cluster shape classification scheme can be generalized to recognize norm induced shells [1]. To increase the stability of previously learned clusters, an ART-centric scheme could be adopted and the reference position updating portion of the algorithm removed. Finally, we are currently expanding the work to include temporal labels and heuristics.

References

[1] J. Bezdek, J. Keller, R. Krishnapuram, N. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer, 1999.

[2] I. Sledge, J. Huband, J. Bezdek, "(Automatic) Cluster Count Extraction from Unlabeled Datasets", *IEEE Proc., ICNC*, 2008.

[3] G. Carpenter, "Competitive Learning: From Interactive Activation to Adaptive Resonance", *Cognitive Science*, vol. 11, 1987.

[4] B. Fritzke, "A Growing Neural Gas Network Learns Topologies", *Advances in Neural Inform. Processing Sys.*, vol. 7, 1994.

[5] B. Dai, et al., "Adaptive Clustering for Multiple Evolving Streams", *IEEE Trans. Knowledge Eng.*, vol. 18, 2006.

[6] P. Hore, L. Hall, D. Goldgof, "Creating Streaming Iterative Soft Clustering Algorithms", *IEEE Proc., NAFIPS*, 2007.

[7] T. Moon, "Temporal Pattern Recognition using Fuzzy Clustering", *IEEE Proc., World Congress on CI*, 1994.

[8] S. Policker, A. Geva, "Non-stationary Signal Analysis using Temporal Clustering", *IEEE Proc., Neur. Net. for Sig. Proc.*, 1998.

[9] M. Berg, M. Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry*, Springer-Verlag, New York, 2000.

[10] L. Khachiyan, M. Todd, "On the Complexity of Approximating the Maximal Inscribed Ellipsoid for a Polytope", *Math. Prog.*, 1993.

[11] J. Bezdek, R. Hathaway, "VAT: A Tool for Visual Assessment of (cluster) Tendency", *IEEE Proc., IJCNN*, 2002.

[12] I. Sledge, et al., "Temporal Activity Analysis", *Proc., AAAI*, 2008.

[13] J. Keller, I. Sledge, "A Cluster By Any Other Name", *IEEE Proc., NAFIPS*, 2007.

[14] I. Sledge, J. Keller, G. Alexander "Emergent Trend Detection in Diurnal Activity", *IEEE Proc., EMBS/BMES*, 2008.