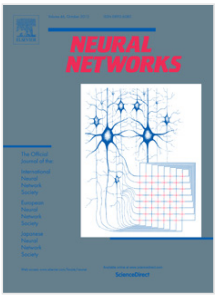
Accepted Manuscript



Growing Neural Gas approach for obtaining homogeneous maps by restricting the insertion of new nodes

Yuri Quintana-Pacheco, Daniel Ruiz-Fernández, Agustín Magrans-Rico

PII: S0893-6080(14)00006-9
DOI: http://dx.doi.org/10.1016/j.neunet.2014.01.005
Reference: NN 3281

To appear in: *Neural Networks*

Received date: 23 February 2013
Revised date: 28 December 2013
Accepted date: 9 January 2014

Please cite this article as: Quintana-Pacheco, Y., Ruiz-Fernández, D., & Magrans-Rico, A. Growing Neural Gas approach for obtaining homogeneous maps by restricting the insertion of new nodes. *Neural Networks* (2014), http://dx.doi.org/10.1016/j.neunet.2014.01.005

Growing Neural Gas approach for obtaining homogeneous maps by restricting the insertion of new nodes

Yuri Quintana-Pacheco[b], Daniel Ruiz-Fernández[a,*], Agustín Magrans-Rico[b]

[a] *Department of Computing Technology, University of Alicante, Alicante, Spain*
[b] *IBIS Research Group, University of Alicante, Alicante, Spain*

Abstract

The Growing Neural Gas model is used widely in artificial neural networks. However, its application is limited in some contexts by the proliferation of nodes in dense areas of the input space. In this study, we introduce some modifications to address this problem by imposing three restrictions on the insertion of new nodes. Each restriction aims to maintain the homogeneous values of selected criteria. One criterion is related to the square error of classification and an alternative approach is proposed for avoiding additional computational costs. Three parameters are added that allow the regulation of the restriction criteria. The resulting algorithm allows models to be obtained that suit specific needs by specifying meaningful parameters.

Key words: Growing Neural Gas, prototype proliferation, self-organizing model, topology preservation

1. Introduction

Topology-preserving clustering methods can find homogeneous groups of data points in a given multidimensional data set while simultaneously preserving the topological structure of the dataset. Thus, the points in the input space, which is generally multidimensional, are represented as points in a space with a low number of dimensions, usually two or three, where the distances between

*Corresponding author. E-mail addresses: druiz@dtic.ua.es

the points in this space correspond to the dissimilarities between the points in the original space. These methods have been used in applications for visualizing the structure of multidimensional data and for the dimensionality reduction of datasets (Duda et al., 2000).

Self-organizing maps (SOMs) (Kohonen et al., 2001) are artificial neural network models that are used widely. From a clustering perspective, the clusters can be identified based on the topological coordinates of the neurons. The production of a SOM includes a training phase when the map is constructed using the elements of the input space via a competitive process (vector quantization). During this process, the neurons are adapted to capture the structure of the input space, which is reflected by the spatial relationships in the map.

Major disadvantages of SOMs in many applications are that the network size and structure type should be predetermined. Numerous alternatives have been proposed to overcome this problem, mainly by using incremental models. The Incremental Grid Growing (Blackmore and Miikkulainen, 1993) method adds new neurons o the border of the map and an expansion factor is used to control the growth process, which is similar to the Growing SOM (Alahakoon et al., 2000) method. In the Growing Grid (Ayadi et al., 2007) method entire rows and/or columns are added to the map during training, which are based on the calculated measures for each neuron. The Hypercubical SOM (Bauer and Villmann, 1997) method is an extension of Growing Grid that allows growth in more than two dimensions. In the Growing Cell Structures (Fritzke, 1993) method, growth occurs around the most active neurons. The Growing Neural Gas (GNG) (Fritzke, 1995) (Martinetz, 1993) method combines the idea of a Neural Gas (Martinetz and Schulten, 1991) with competitive Hebbian learning rules (Martinetz, 1993). This method adds new neurons based on local statistical measures, which are collected during the adaptation process.

Growing models have been used widely in recent years for different applications, but mainly for clustering or topology learning. In some studies, the GNG algorithm has been improved or adapted to different applications (García-Rodríguez et al., 2012). In (Furao et al., 2007), an incremental learning GNG

model was proposed for handling non-stationary problems and this issue was also addressed in (Qin and Suganthan, 2004). In (Hebboul et al., 2011), a two-layer design was proposed for increasing robustness against noise. In (do Rego et al., 2010), GNG was extended using the concept of triangular mesh faces for use as a surface reconstruction method.

An undesirable behavior of models generated using the GNG algorithm is the proliferation of nodes in high density areas of the dataset. This node proliferation occurs because there is no criterion to stop the insertion of nodes in areas that are sufficiently well represented. The proliferation of nodes results in an unbalanced model with less represented wider areas and the growth model also has an increased execution time.

One strategy for controlling prototype proliferation is the use of magnification control. Magnification describes the relationship between the data and weight vector density. In (Villmann and Claussen, 2006), this approach was applied to fixed network size algorithms, SOM and NG, which allowed the generation of a model with a predefined magnification value.

Sparse clustering is based on the fact that only a small fraction of features is relevant during class discovery and several studies have investigated the problems associated with data that have these characteristics (Villmann et al., 2010). For example, Witten and Tibshirani (Witten and Tibshirani, 2010) propose a framework that can be applied to any similarity-based clustering technique for feature selection during sparse clustering (reducing the complexity of the model), although they used k-means clustering

In (Satizábal et al., 2008), a method was proposed for avoiding prototype proliferation based on quantization error control, although this method required the definition of various parameters based on previous knowledge of the scale of datasets. This also applied to the method proposed in (Marsland et al., 2002), where new nodes can be added at any time during the learning process, which are positioned based on the input and the current winning node, rather than adding them where the accumulated error is maximized, which is the case with GNG.

2. Growing Neural Gas

The GNG algorithm can learn the topological relationships for a given dataset using Hebbian learning rules (Fritzke, 1995). The main concept employed by this method is to add new nodes to an initially small network by evaluating local statistical measures collected during the early stages of adaptation.

The model generated by the algorithm is a graph or network, which has the following components.

- A set of nodes $\mathcal{V}$ (neurons). Each node $v \in \mathcal{V}$ has a reference vector $w_v \in \mathbf{R}^n$, which can be regarded as its position in the input space.
- A set of edges $\mathcal{A}$ between pairs of nodes. These edges have no associated weights and they are used to define the topological structure.

The GNG algorithm analyzes the signals in the input space according to a probability distribution $P(\xi)$, as follows.

1. Start with two nodes in random positions w_a and w_b in $\mathbf{R}^n$.
2. Generate an input signal ξ.
3. Find the nearest (s_1) and the second-nearest (s_2) nodes.
4. Increment the age of the edges from s_1.
5. Add the squared distance between the input signal and the nearest node (s_1) to a local counter variable:

$$\Delta error(s_1) = \|w_{s_1} - \xi\|^2 \tag{1}$$

6. Move s_1 and its direct topological neighbors toward ξ by fractions of the total distance (ϵ_b and ϵ_n, respectively):

$$\Delta w_{s_1} = \epsilon_b(\xi - w_{s_1}) \tag{2}$$

$$\Delta w_n = \epsilon_n(\xi - w_n) \qquad \text{for all direct neighbors } n \text{ of } s_1 \tag{3}$$

7. If s_1 and s_2 are connected, set the age of the respective edge to zero. If there is no connection, create one.
8. Remove connections with an age greater than a_{max}. Remove disconnected nodes.
9. After every λ iterations, insert a new node as follows.
   - Determine the node (q) with the maximum accumulated error.
   - Insert a new node (r) halfway between q and its neighbor f with the largest error value:

$$w_r = 0.5(w_q + w_f) \tag{4}$$

   - Connect node r with q and f, and remove the original edge between q and f.
   - Decrease the error variables of q and f (multiply them by a constant α). Initialize the error variable of r with the new value of the error variable of q.
10. Decrease all of the error variables (multiply them by a constant d).
11. If a stop criterion (e.g., some performance measure or net size) is not met, return to step 2.

The adaptation to the inputs (step 6) leads to a general movement of the nodes to the areas of the space where the input signals are located. The insertion of edges (step 7) between the nearest and second nearest nodes with respect to the input signal is part of the topological structure construction process.

Edge removal (step 8) is necessary to eliminate edges that have lost their topological significance during the movement of the connecting nodes. This process is based on the aging of the edges of the winning node (step 4), which is combined with the strengthening of the edges between the nearest and the second nearest nodes (step 7).

The accumulation of errors (step 5) during the fitting process helps to identify nodes that are located in the areas of the input space where signal mapping

produces the greatest errors. New nodes are created in these areas to reduce these errors.

3. GNG with insertion restrictions

The approach proposed in this study is based on two concepts employed by the method presented in (Furao et al., 2007). First, new nodes are inserted into the model when input signals are located in undiscovered areas. A coverage threshold is maintained in every node and a new node is created if the input signal is outside the coverage of the map. Second, a cooling schedule is used during movement adaptation for the winning neurons and their neighbors (equations 2 and 3), instead of using fixed parameters. The fraction that nodes move is based on the number of times each neuron has won. Thus, neurons that have won more times are adapted by a smaller fraction because they have better locations (a similar idea is discussed in (Schleif et al., 2007) in the context of supervised LVQ).

Our proposed method limits node proliferation by applying three restrictions while inserting new nodes using the GNG algorithm. Each restriction uses a different approach, which tries to maintain some degree of homogeneity in the resulting model. This is achieved by introducing a new parameter for each constraint, which controls the proportion of local or temporary measures used for the criterion and the global maximum.

These restrictions make the GNG algorithm more robust with respect to the parameter λ, which controls the insertion of new nodes. This parameter has a strong influence on the number of nodes in the resulting model because although GNG has a node removal mechanism, its effect on the model size is quite small. Therefore, by applying the constraints, the number of nodes will be related more strongly to the new parameters associated with the proposed restrictions than λ. These new parameters provide a clear interpretation of the characteristics of the model that needs to be obtained and they are designed to take values between 0 and 1.

3.1. Restriction based on the radius of nodes

A measure that may reflect the representation degree in an area of a model is the radius of the nodes, which is defined as the average distance between a node and each of its neighbors:

$$r_q = \frac{1}{|\mathcal{V}_q|} \sum_{v \in \mathcal{V}_q} \|w_q - w_v\| \tag{5}$$

where $\|\mathcal{V}_q\|$ is the number of elements in the set of the neighbors of node q.

From this definition, the goal is to obtain a model where the radius of the nodes is within a range of values that produces a relatively uniform lattice. This goal is achieved by preventing the insertion of new nodes in areas where the radius of a node is disproportionately lower than the larger radius in the model. Thus, insertion is performed around the node q if it holds that:

$$\frac{r_q}{r_{max}} > \varphi \tag{6}$$

where r_q is the radius of node q, r_{max} is the larger radius of a node in the model, and φ is the parameter added to the algorithm to control the insertion of new nodes.

3.2. Restriction based on the density of nodes

The density of a node is a measure that can be used to control the proliferation of nodes by avoiding the insertion of new nodes in high density areas. In general, the density of a node is related to the number of elements in the dataset accumulated around a node. Thus, the density is high if many elements are present near a node, but low in the opposite case. In (Furao et al., 2007), a definition of density was proposed that considers the number of times a node has won and the relationship between a node with its neighbors. To determine the density of a node, the radius of a node i is calculated (equation 5) first, which is defined as the *point* of a node:

$$p_i = \begin{cases} \dfrac{1}{(1+r_i)^2} & \text{if node } i \text{ is the winner} \\ 0 & \text{if node } i \text{ is not the winner} \end{cases} \tag{7}$$

The *point* values are calculated only for the winner node, so the *point* of the winner changes during one iteration whereas it remains identical for the others.

The accumulated points a_i are calculated as the sum of the points for node i during a learning period:

$$a_i = \sum_{k=1}^{t} \left(\sum_{l=1}^{\lambda} p_i \right) \tag{8}$$

where t is the number of learning periods, which is calculated as:

$$t = \frac{N}{\lambda} \tag{9}$$

where N is the number of input signals. Finally, the density of node i is calculated as:

$$D_i = \frac{1}{T} a_i \tag{10}$$

where T represents the number of learning periods during which a_i is greater than 0.

Given this definition of density, the aim is to maintain a level of density homogeneity among the nodes of the map. Thus, a restriction is added to the node insertion process, as follows:

$$\frac{D_q}{D_{max}} > \gamma \tag{11}$$

where D_q is the density of the node q, D_{max} is the density of the node with a higher density, and γ is the parameter that controls the maximum allowable density ratio, thereby restricting the insertion of new nodes in the model.

3.3. Restriction based on the squared error

A feature that contributes to the proliferation of nodes is that the GNG algorithm lacks a criterion for stopping growth based on the quality of the model. A quality measure that can be applied to clustering algorithms is the squared error in the classification of all elements in a given dataset, i.e.:

$$e = \sum_{\xi \in \mathcal{D}} \min_{v \in \mathcal{V}} \|\xi - v\|^2 \tag{12}$$

where $\mathcal{D}$ is the input dataset and $\mathcal{V}$ is the set of nodes in the model. Thus, a possible method for stopping growth could be the insertion of new nodes when the quality of the model is below a predetermined value.

The datasets shown in Figure 1 were used to study the behavior of the mean squared error for different values of the parameter λ (Figure 2, first column). It can be seen that the value of the squared error decreases as the learning process continues. The curves that correspond to the higher values of λ are above the curves that corresponding to the lower values of λ. This is because the model grows faster with higher values of λ and the models with more nodes (figure 2, third column) are more representative, thus the squared error value is lower.

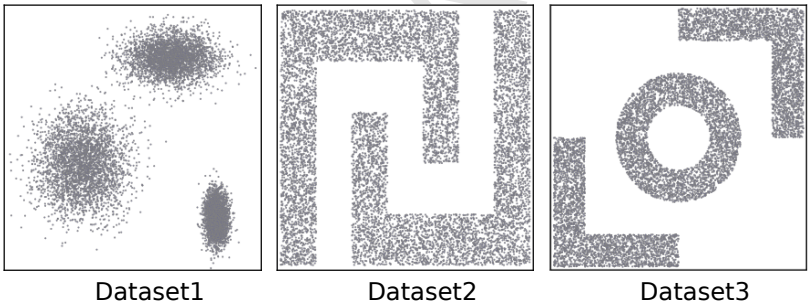
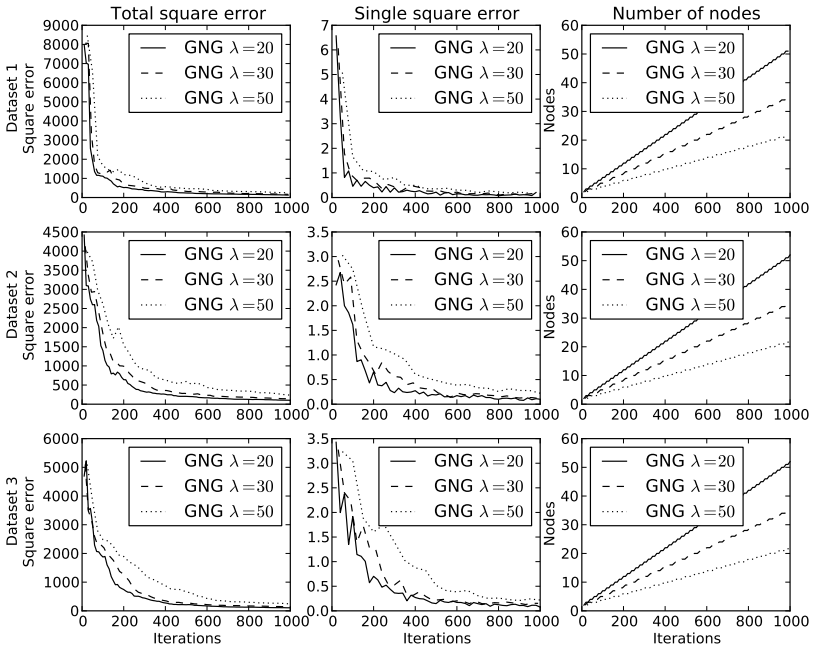


Figure 1: Artificial datasets used in the experiments.

In all cases, the model reached stability after a stage of accelerated learning where the mean squared error decreased significant. Thus, at some point, an increase in the number of nodes does not produce a significant decrease in the mean squared error. Indeed, a point is reached where there are no significant differences between the curves that correspond to different values of λ.

From this analysis, an insertion requirement analogous to the above criteria can be set. Thus, to insert a new node in the model, the ratio between the squared error at the time of insertion (e_j, square error at iteration j) and the larger squared error already reached during the learning (e_{max}) must be greater



GNG parameters ($\epsilon_b = 0.2$, $\epsilon_n = 0.01$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$)

Figure 2: Behavior of the total squared error, single squared error, and the number of nodes in the model for different λ values.

than a new parameter, σ, which is expressed as:

$$\frac{e_j}{e_{max}} > \sigma \tag{13}$$

However, this approach has a major problem because the calculation of the squared error is computationally highly expensive. Thus, the use of an alternative measure related to the mean squared error is proposed, which does not incur high a computational cost. There is a significant relationship between the squares error and the *single squared error* (figure 2, center column), i.e., the squared distance between an element in the input space and the nearest node of the model:

$$\hat{e}_j = \|w_{s_1} - \xi\|^2 \tag{14}$$

where $\hat{e}_j$ is the single squared error in iteration j, ξ is the signal in the input space, and w_{s_1} is the reference vector of the winning node s_1. This relationship is supported because when the models fits the data, a better representation is achieved using growth or learning and the new elements in the input dataset will generally be closer to a specific node. However, it is clear that these values can be quite different in many cases, especially in the presence of noise or when recognizing new areas. Given this potential variability, we used the average squared error ($\bar{E}_k$) between two insertion moments (k y $k+1$):

$$\bar{E}_k = \frac{1}{\lambda} \sum_{j=k\lambda}^{(k+1)\lambda} \hat{e}_j \tag{15}$$

Table 1 shows the correlation coefficients of the average single squared error between two points of insertion and the total squared error:

$$r = \frac{1}{t-1} \sum_{i=1}^{t} \frac{\bar{E}_{ki} - \bar{\bar{E}}_k}{S_{\bar{E}_k}} \frac{e_i - \bar{e}}{S_e} \tag{16}$$

where

$$\bar{x} = \frac{1}{t} \sum_{i=1}^{t} x_i \tag{17}$$

and

$$S_x = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t}(x_i - \bar{x})^2} \tag{18}$$

Table 1: Correlation between the total squared error and average single squared error.

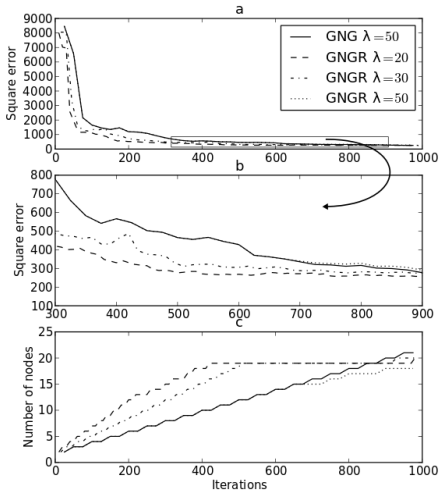
Datasets	$\lambda = 20$	$\lambda = 30$	$\lambda = 50$
Dataset 1	0.9841837	0.9936787	0.9953250
Dataset 2	0.9847602	0.9798915	0.9787587
Dataset 3	0.9669757	0.9925371	0.9731519

The high correlation coefficients support the feasibility of using the simple squared error between two insertion moments to estimate the behavior of the total squared error at a specific iteration. The advantage of using this estimation method is that it incurs virtually no additional computational load, because it calculates the distance between an element in the input space and the node of the model that is nearest in the GNG algorithm. Finally, the condition that needs to be imposed on the insertion of a new node is:

$$\frac{\bar{E}_k}{\bar{E}_{max}} > \sigma \tag{19}$$

where σ is the parameter introduced to avoid insertion when the single average squared error has not decreased significantly, thereby preventing the unnecessary growth of the model.

Figure 3 compares the results obtained using the GNG algorithm with and without the restriction based on the squared error. Figure 3a shows that in the beginning, the behavior of both algorithms is the same because the constraint is not met and the same pattern is produced. At some point, the constraint is met, as shown in Figure 3c, where the growth varies with the number of nodes in the model. This figure also shows that the models obtained using the algorithm with the restriction have similar sizes and mean squared error values (Figure 3b), regardless of the value of λ.



GNG parameters ($\epsilon_b = 0.2$, $\epsilon_c = 0.01$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$)
Restriction parameter ($\sigma = 0.2$)

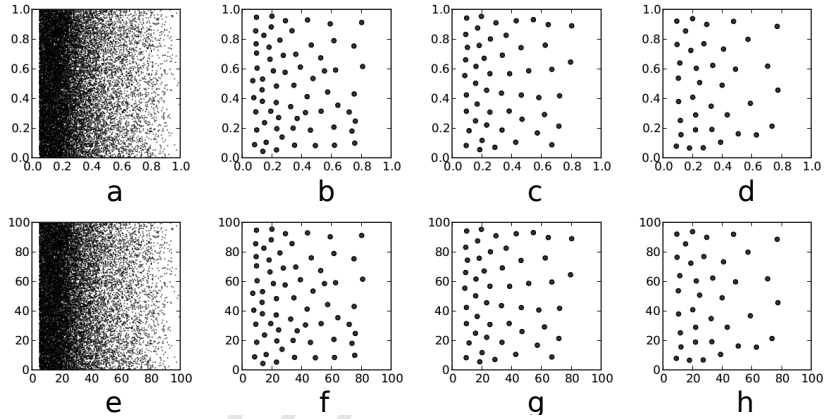
Figure 3: Behavior of the squared error (a and b) and the number of nodes (c) for the GNG algorithm with (GNGR) and without (GNG) insertion restriction based on the squared error for the experimental datasets.

3.4. Computational complexity

The GNG algorithm has an approximate time complexity of $\mathcal{O}(NV)$ given that the nearest node in the model has to be found for each input sample, and a space complexity of $\mathcal{O}(V)$ to store information related to each node (N is the number of input samples and V is the number of nodes in the model). With the proposed modifications, the space complexity is affected mainly by adding new information to each node based on the criteria used by the proposed restrictions. The time complexity is affected mainly by operations related to the restrictions, including updating the criteria values of nodes and updating the maximum criteria values. Thus, the modifications proposed in this study do not significantly affect the overall time and space complexity of the original GNG algorithm.

4. Results

Figure 4 shows the results obtained using two experimental datasets. Both datasets (Figure 4a and Figure 4e) comprise two-dimensional points, which are distributed uniformly on the y axis with a declining distribution on the x axis. The only difference is that the points in Figure 4e have a higher scale than those in Figure 4a. Figures 4b, 4c, and 4d show the results obtained using the proposed algorithm with increasing values of the restrictions parameters for the first dataset. These changes produced more homogeneous maps as the restrictions became more stringent and the prototype proliferated less in the denser regions of the first dataset.



Shared parameters ($\lambda = 500$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$)

Figure 4: Models generated using the GNG algorithm with the proposed restrictions for two different datasets. a-) Dataset with the range (0,1) and e-) dataset with the range (0, 100). The models in the top row were obtained using the dataset at the top: b-) using $\varphi = 0$, $\gamma = 0$, $\sigma = 0$; c-) using $\varphi = 0.15$, $\gamma = 0.15$, $\sigma = 0.15$; and d-) using $\varphi = 0.25$, $\gamma = 0.25$, $\sigma = 0.25$. The models in the bottom row were obtained using the dataset at the bottom: f-) using $\varphi = 0$, $\gamma = 0$, $\sigma = 0$; g-) using $\varphi = 0.15$, $\gamma = 0.15$, $\sigma = 0.15$; and h-) using $\varphi = 0.25$, $\gamma = 0.25$, $\sigma = 0.25$.

Figures 4f, 4g, and 4h show that similar results were obtained for the second dataset when using the exact same parameters in the proposed algorithm. This highlights a strength of the proposed algorithm because no prior knowledge of the data scale is required to specify the parameters of the algorithm (whereas other methods require the data scale, e.g., (Satizábal et al., 2008) and (Marsland

et al., 2002)).

Figure 5 shows a comparison of the effects of the data scale with the proposed method and two other methods that also control the insertion of new nodes to avoid prototype proliferation. The first row shows the results obtained using the proposed method (Figure 5b), those obtained using the method presented in (Satizábal et al., 2008) (Figure 5c), and those presented in (Marsland et al., 2002) (Figure 5d). These results were obtained using the dataset shown in Figure 5a. The models had relatively sparse nodes with all three methods.

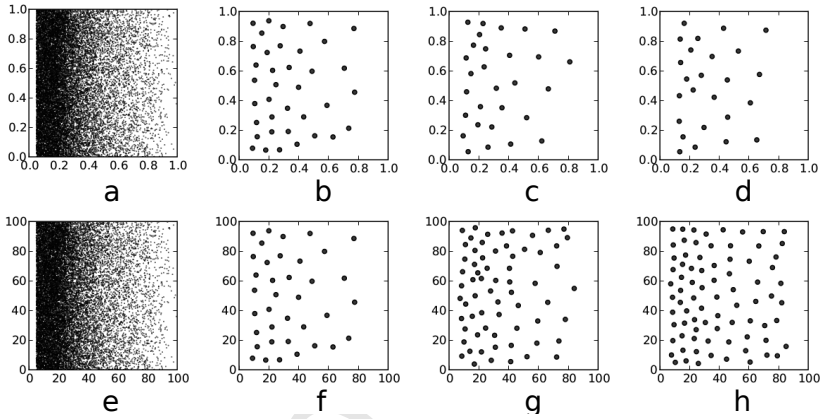
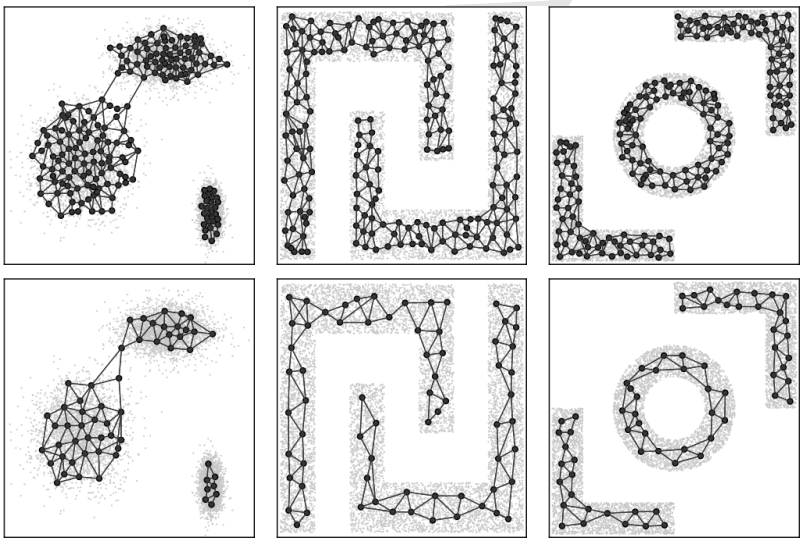


Figure 5: Comparison of the effect of the data scale using different GNG-based algorithms. a-) Dataset with the range (0,1) and e-) dataset with the range (0, 100). The models in the top row were obtained using dataset (a) and those in the bottom row were obtained using dataset (e): b, f-) algorithm GNG with proposed restrictions ($\lambda = 500$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$, $\varphi = 0.25$, $\gamma = 0.25$, $\sigma = 0.25$), c, g-) the algorithm proposed in (Satizábal et al., 2008)($\lambda = 500$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$, $qE = 0.1$, $sp = 0.0.75$, $h = 0.25$), and d, h-) the algorithm proposed in (Marsland et al., 2002) ($h_0 = 1$, $\alpha_b = 1.05$, $\alpha_n = 1.05$, $\tau_b = 3.33$, $a_T = 0.85$).

In the figure, the second row shows the results obtained using the same methods with identical parameter values, except they are applied to the dataset shown in Figure 5e. This dataset is similar to Figure 5a but it has a higher scale. The results obtained using the proposed algorithm had very similar structures and numbers of nodes with both datasets (Figure 5f). By contrast, the models obtained using the algorithm proposed in (Satizábal et al., 2008) (Figure 5g) and in (Marsland et al., 2002) (Figure 5h) were significantly different, where

the results produced using both models had a higher number of nodes. The difference in the number of nodes occurred because both algorithms use parameters to regulate the insertion of new nodes that are sensitive to the data scale (qE and sp for the method proposed in (Satizábal et al., 2008) and a_T for the method proposed in (Marsland et al., 2002)). Thus, prior knowledge is required of the data scale and the experimental set up to specify appropriate values for these parameters.

Figure 6 compares the results obtained using the GNG algorithm with and without the proposed restrictions. In the experiments, we used the same values for the parameters in both algorithms, thereby ensuring that they had similar basic conditions.



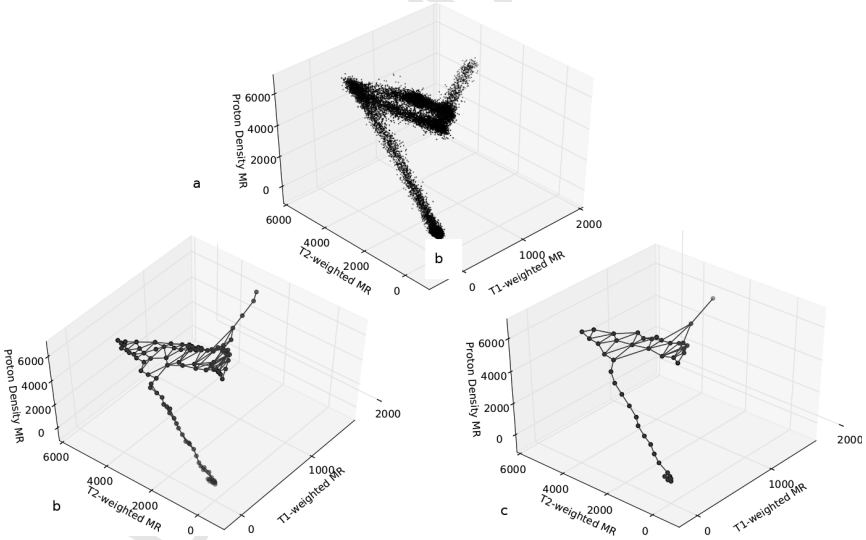
Shared parameters ($\lambda = 50$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$)
GNG parameters ($\epsilon_b = 0.2$, $\epsilon_n = 0.01$)
Restriction parameters ($\varphi = 0.15$, $\gamma = 0.15$, $\sigma = 0.15$)

Figure 6: Comparison of the models generated using the GNG (top) algorithms and the GNG algorithms with the proposed restrictions (bottom).

The results show that there were clear differences in the number of nodes in the models obtained using each algorithm for each dataset, where much smaller models were obtained using the algorithm with the proposed restrictions. Both

algorithms made the same number of attempts to insert new nodes around the node with the highest accumulated error because the datasets and the value of λ were the same. The proposed algorithm also adds nodes as part of the accelerated learning process, but the models obtained using the proposed algorithm had fewer nodes (with a significant representative quality) because of the restrictions during the insertion phase, and they had homogeneous lattice patterns.

The proposed algorithm was also tested with a real-world dataset. Simulated magnetic resonance images (MRIs) of SBD's anatomical model of a human brain with mild MA lesions were obtained from the BrainWeb custom MRI simulation server (BrainWeb, 2013). Three images were obtained with T_1, T_2, and PD pulse sequences. The feature space used for training was defined as 3D vectors with x, y, and z coordinates, which corresponded to T_1, T_2, and PD image intensities, respectively. Figure 7a shows the spatial distribution of this dataset.



Shared parameters ($\lambda = 200$, $a_{max} = 50$, $\alpha = 0.5$, $d = 0.9$)
GNG parameters ($\epsilon_b = 0.2$, $\epsilon_n = 0.01$)
Restriction parameters ($\varphi = 0.15$, $\gamma = 0.15$, $\sigma = 0.15$)

Figure 7: Comparison of the models generated the GNG algorithm (b) and the GNG algorithm with the proposed restrictions (c) based on the BrainWeb dataset (a).

Figures 7b and 7c show the models generated using the GNG algorithm and

the GNG algorithm with the proposed restrictions, respectively, based on the BrainWeb dataset. In agreement with our previous experiments, the application of the proposed restrictions to the GNG produced more homogeneous models by avoiding prototype proliferation in denser areas.

5. Conclusions

In this study, we proposed a modified version of the algorithm GNG where the main objective is to avoid prototype proliferation in dense areas of the input space. This goal is achieved by adding three restrictions on the insertion of new nodes, each of which aims to maintain a homogeneous level based on the criteria used. The restriction criteria are related to the radius, density, and squared error. For the squared error-related criterion, an estimate is used to avoid additional computational costs. Each restriction is controlled by a different parameter, which regulates the required level of homogeneity.

The proposed algorithm allows the production of models that meet specific needs by setting appropriate values for the parameters associated with the restrictions. If the restrictions are relaxed, the model produced will be similar to the respective GNG model.

References

Alahakoon, D., Halgamuge, S., and Srinivasan, B. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3):601 –614.

Ayadi, T., Hamdani, T., Alimi, A., and Khabou, M. (2007). 2IBGSOM: interior and irregular boundaries growing self-organizing maps. In *Sixth International Conference on Machine Learning and Applications, 2007 (ICMLA 2007)*, pages 387 –392.

Bauer, H.-U. and Villmann, T. (1997). Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8(2):218 –226.

Blackmore, J. and Miikkulainen, R. (1993). Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map. In *Proceedings of the IEEE International Conference on Neural Networks (San Francisco, CA)*, pages 450–455. Piscataway, NJ: IEEE.

Brainweb (last access July 2013) Simulated Brain Database, http://www.bic.mni.mcgill.ca/brainweb/.

do Rego, R. L. M. E., Araújo, A. F. R., Neto, F. B. L. (2010). Growing self-reconstruction maps. *IEEE Transactions on Neural Networks*, 21:211–223.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Fritzke, B. (1993). Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7:1441–1460.

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press.

Furao, S., Ogura, T., and Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, 20(8):893 – 903.

García-Rodríguez, J., Angelopoulou, A., García-Chamizo, J. M., Psarrou, A., Escolano, S. O., and Giménez, V. M. (2012). Autonomous growing neural gas for applications with time constraint: Optimal parameter estimation. *Neural Networks*, 32(0):196 – 208.

Hebboul, A., Hacini, M., and Hachouf, F. (2011). An incremental parallel neural network for unsupervised classification. In *Seventh International Workshop on Systems, Signal Processing and their Applications (WOSSPA), 2011* , pages 400 –403.

ICBM: International Consortium for Brain Mapping, http://www.loni.ucla.edu/ICBM/.

Kohonen, T., Schroeder, M. R., and Huang, T. S., editors (2001). *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition.

Marsland, S., Shapiro, J., Nehmzow, U. (2002). A self-organizing network that grows when required. *Neural Networks*, 15:1041–1058.

Martinetz, T. and Schulten, K. (1991). A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks*, I:397–402.

Martinetz, T. M. (1993). Competitive Hebbian learning rule forms perfectly topology preserving maps. In *Proceedings of ICANN '98*, pages 427–434, Amsterdam. Springer.

Martinetz, T.M., Berkovich, S.G. and Schulten, K.J. (1993). 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569.

Qin, A. K., Suganthan, P. N. (2004). Robust growing neural gas algorithm with application in cluster analysis. *Neural Networks*, 17:1135–1148.

Rauber, A., Merkl, D., and Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13:1331–1341.

Satizábal, H. F., Pérez-Uribe, A., and Tomassini, M. (2008). Prototype proliferation in the growing neural gas algorithm. In *Proceedings of the 18th International Conference on Artificial Neural Networks, Part II*, ICANN '08, pages 793–802, Berlin, Heidelberg. Springer-Verlag.

Schleif F.-M., Hammer B., and Villmann T. (2007). Margin-based active learning for LVQ networks. *Neurocomputing*, 70(7-9):1215–1224.

Villmann, T., Claussen J. C. (2006). Magnification Control in Self-Organizing Maps and Neural Gas. *Neural Computation*, 18(2):446–469.

Villmann, T., Schleif F.-M., and Hammer, B. (2010). Sparse representation of data. In *Proceedings of ESANN 2010*, 225–234.

Witten, D. M., Tibshirani, R. (2010). A Framework for Feature Selection in Clustering. *Journal of the American Statistical Association*, 105:490.