Assessing Satellite Image Time Series Clustering Using Growing SOM

Rodrigo de Sales da Silva Adeu[1], Karine Reis Ferreira[1], Pedro R. Andrade[1], and Lorena Santos[1]

[1] National Institute for Space Research (INPE),
Astronautas Avenue 1758, 12227-010,
São José dos Campos – São Paulo – Brazil.
http://www.inpe.br/
{rodrigo.adeu, karine.ferreira, pedro.andrade, lorena.santos}@inpe.br

Abstract. Mapping Earth land use and cover changes is crucial to understand agricultural dynamics. Recently, analysis of time series extracted from Earth observation satellite images has been widely used to produce land use and cover information. In time series analysis, clustering is a common technique performed to discover patterns on data sets. In this work, we evaluate the Growing Self-Organizing Maps algorithm for clustering satellite image time series and compare it with Self-Organizing Maps algorithm. This paper presents a case study using satellite image time series associated to samples of land use and cover classes, highlighting the advantage of providing a neutral factor (called spread factor) as a parameter for GSOM, instead of the SOM grid size.

Keywords: Growing Self-Organized Map · Land Use and Cover · Machine Learning

1 Introduction

Technologies and methods of remote sensing and digital image processing play a crucial role in the identification, mapping, assessment and monitoring of land use and cover changes. In the last years, the use of remote sensing image time series analysis to produce land use and cover information has increased greatly [6]. Bi-temporal or traditional change detection approaches are commonly used to detect complex underlying processes, but time series derived from Earth observation satellite images has been used as an alternative to facilitate this task [10].

In order to classify Earth observation image time series to produce land use and cover change maps, machine learning methods such as Support Vector Machine (SVM) and Random Forest (RF) have been used [11]. Most of these methods are based on supervised learning algorithms, requiring a training step that uses labeled land use and cover samples. A crucial challenge on this task is the selection of representative samples to obtain a good classification accuracy.

To better select land use and cover samples from satellite image time series, Santos et al. [13] propose a method based on Self-Organizing Maps (SOM) neural network [8] for time series clustering. Such method estimates the land use

and cover samples quality from the satellite image time series clusters created by SOM.

Despite its advantages, SOM has a characteristic that limits its potential. It uses a predefined and fixed network architecture in terms of number and arrangement of neural processing elements. Simulations have to be run several times on different network sizes to find an appropriate network structure [5]. This work aims to contribute to land use and cover research area by validating an approach that avoids this additional parameter. Growing SOM (GSOM) was originally proposed to address the SOM characteristic of predetermining the map size [1]. This paper evaluates GSOM as an alternative to traditional SOM for satellite image time series clustering.

2 Self-Organizing Map (SOM)

A Self-Organizing Map (SOM) is an unsupervised neural network. SOM maps a high dimensional space onto a low-dimensional space preserving its neighbourhood topology. SOM is composed by input and output layers, with the latter generally being a two-dimensional grid.

Each element of a grid is called neuron. An important property of SOM is that the neurons are organized in a way that they maintain a similar neighbourhood, that is, neurons that have similar characteristics are close in the output layer. Figure 1 shows the structure of SOM.

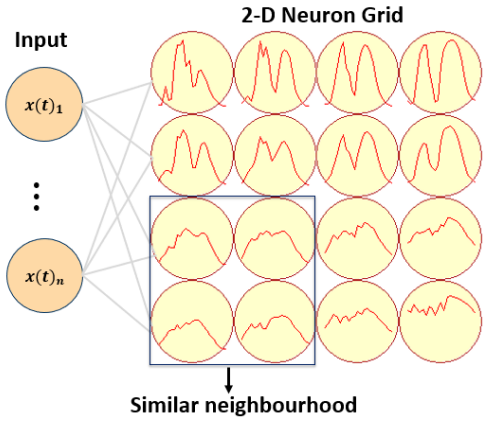

Fig. 1. SOM Structure

Each neuron j has a n-dimensional weight vector $w_j = [w_1, \ldots, w_n]$ associated to it. An input $x(t) = [x(t)_1, \ldots, x(t)_n]$ is associated to most similar neuron to it through distance metrics, as Euclidean distance. The distance D_j is computed between a input vector and each neuron j for all the neurons in the output layer (equation 1).

$$D_j = \sum_{i=1}^{N} \sqrt{(x(t)_i - w_j)^2}. \quad (1)$$

Once we have with all distances between a input and all neurons, the minimum distance is determined the Best-Matching-Unit (BMU), i.e. the neuron d_b with weight vector closer to $x(t)$ (equation 2):

$$d_b = min\left\{D_1, \ldots, D_j\right\}. \quad (2)$$

The BMU and its neighbours within a radius r must be updated. The weights are adjusted to increase the similarity with the input vector, the update is given by:

$$w_j(t) = w_j(t) + \alpha(t) \times h_{b,j}(t)[x(t)_i - w_j(t)], \quad (3)$$

In equation 3, $\alpha(t)$ is the learning rate, set as $0 < \alpha(t) < 1$ and $h_{b,j}$ is a neighbourhood function. The SOM mapping ends when all input vectors are presented to output layer. During each time, the $\alpha(t)$ must be reduced and the neighbourhood function reduce the radius of the neighborhood. There are several ways to reduce the value of $\alpha(t)$ and the radius of the neighbours. They can be found in [9].

3 Growing SOM

Growing SOM (GSOM) is an alternative to traditional SOM for satellite image time series clustering. It was originally proposed to address the SOM requirement of predetermining the map size [1]. SOM attempts to fit a data set into a predefined structure by self-organizing its node weights as well as possible within its fixed borders, while in GSOM the network borders are expandable, generating new nodes whenever needed to expand the network outwards.

3.1 GSOM Algorithm

The GSOM is parameterized by a *spread factor*, a data dimensionality neutral factor. It can be used as a controlling measure for generating maps with different sizes, without previous knowledge about the dataset number of samples or attributes. The GSOM learning algorithm has three phases:

Initial phase: At the initialization phase, GSOM network starts with four neurons with randomly assigned weight vectors. All the initial neurons are boundary nodes and have opportunity to grow. In Figure 2, the four initial neurons are connected with lines and the available positions are shown via dashed circles.

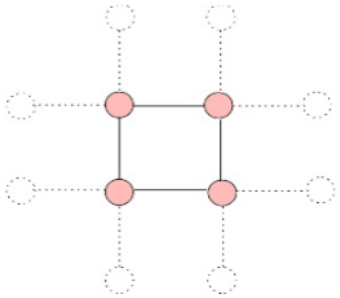

Fig. 2. Initial GSOM with four neurons. Source: [14]

Growing phase: At this phase, the time series data are presented to the network. The weight vector that is closest to the input vector mapped (the winner neuron), is selected based on a distance function. The winner neuron accumulative error is calculated according to a distance function between the input vector and the winner neuron weight vector. This error indicates the distance between the input vector and the weight vector of neurons. When the accumulative error of a neuron exceeds a *growing threshold* (calculated based on the spread factor), and the candidate neuron is on the grid boundary, new neurons are added in the available free positions around the candidate neuron in the grid, as showed on Figure 3. If the winning neuron is not on the grid boundary, the accumulated error of the neighbors are updated according to the winner distance, giving the non-boundary nodes some ability in initiating node growth.

Smoothing phase: In order to fine-tune the weight vectors and to improve the map smoothness, a smoothing phase is applied after growing [14]. No new nodes are added during this phase. The intent is to smooth out any existing quantization error, mainly in the nodes grown at the growing phase latter stages [1]. The smoothing phase stops when the nodes error values in the map become very small.

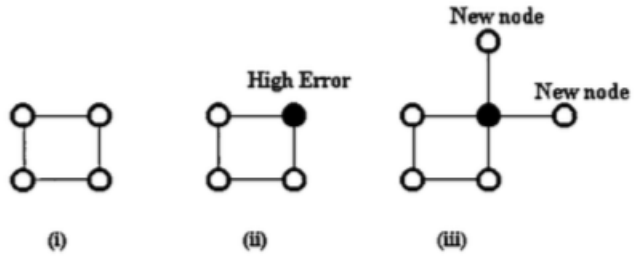

Fig. 3. New node generation from the network boundary. Source: [1]

Depending on the clusters present in the data, the GSOM map generated by this process develops into different shapes. The GSOM shape represents the data grouping, and therefore, such grouping has a better opportunity of attracting the user attention for further investigation [1].

3.2 GSOM customization for satellite image time series

In this paper, we propose minor changes in the GSOM described by Alahakoon et. al. [1]. These changes aim to fit the algorithm to work with satellite image time series.

During the network weights adaptation, the described learning rate states that it needs to be a function that gradually takes higher values as the map grows and the number of nodes becomes larger [1]. The suggested function is described by equation (1) with $R = 3.8$, and $n(t)$ is the number of neurons on a given iteration t.

$$(\frac{1-R}{n(t)}) \tag{4}$$

The use of this learning rate function resulted on a classification accuracy decrease. In order to improve the accuracy, this function was replaced by the function described by equation (2), that gradually takes minor values as the number of iterations increase:

$$e^{(\frac{-iteration}{epochs})}, \tag{5}$$

with *iteration* representing the current iteration number and *epochs* representing the total amount of epochs.

Alahakoon et. al. [1] states that the GSOM starting neighborhood selected for weight adaptation is smaller compared to the SOM, and weight adaptation is carried out with reducing neighborhood until neighborhood is unity. But a function for initializing and reducing the neighborhood influence during the growing and smoothing phase is not provided. To satisfy these requirements, after several tests, we defined the neighborhood influence as showed by Equation (3):

$$e^{(\frac{-d}{2\times\sigma^2})}, \tag{6}$$

$$\sigma = ini \times e^{(\frac{-iteration}{epochs})}, \tag{7}$$

where *ini* is the initial neighborhood influence, *iteration* is the current iteration number, and *epochs* is the total amount of epochs. On all GSOM executions used in this work, the initial neighborhood influence was configured as 0.6.

4 Case study: SOM x GSOM comparative

In order to check the accuracy of GSOM for satellite image time series clustering and to compare it with SOM, we implemented both SOM and GSOM in

Python [12], with the changes proposed in this paper. We then executed a case study using the same time series data set described in Santos et. al. [13]. In this case study, we applied both Python algorithms to this data set and compare the results.

4.1 Data set

The satellite image time series data set used in our case study is shown in Figure 4. This data set is composed by 2215 ground samples of land use and cover classes, including natural vegetation and agricultural, for the Mato Grosso state in Brazil. These samples refer to nine distinct land use and cover classes: (1) Cerrado, (2) Pasture, (3) Forest, (4) Soy-Corn, (5) Soy-Cotton, (6) Soy-Fallow, (7) Soy-Millet, (8) Fallow-Cotton and (9) Soy-Sunflower.

Each sample has a spatial location (latitude and longitude), start and end date that corresponds to agricultural year (from August to September) and the corresponding sample class label. For each sample spatial location, we got the time series associated to that location or pixel from a satellite image collection ordered in time, as shown in Figure 4. In this work, we used satellite images of the MODIS (Moderate-Resolution Imaging Spectroradiometer) sensor of the NASA satellite Terra, including its product MOD13Q1. Each time series has multiple attributes that was generated by EVI, NDVI, NIR and MIR attribute concatenation. These ground samples were collected by [11].

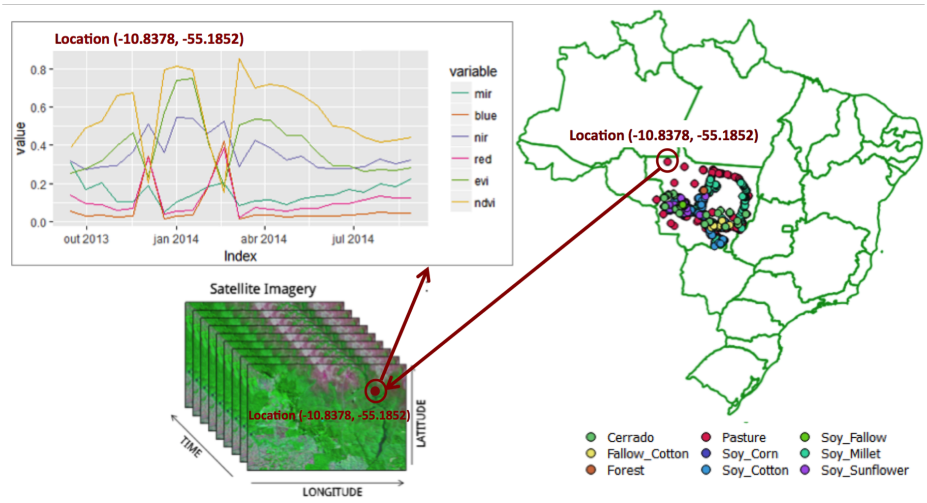

Fig. 4. Satellite image time series data set associated with land use and cover samples. Source: [4]

4.2 Network topology

SOM uses a fixed network architecture in terms of number and arrangement of neural processing elements which have to be predefined. To better estimate the SOM network size, a preliminary study about the number of attributes, classes separability and number of data samples needs to be done. On the other hand, GSOM requires only a spread factor as an input parameter. The spread factor is a neutral number between 0 and 1 that defines how much the network needs to grow.

Figure 5 shows the networks create by SOM (using a fixed 25 x 25 grid) and GSOM (using a spread factor 1 and number of iteration 15; 10 iterations for growing and 5 iterations for smoothing). Both algorithms achieved similar classification accuracy, but with different network topologies. In the SOM case, as the grid size is always prefixed, the cluster distribution can differ, but the grid will always have the same size. On the other hand, in the GSOM case, we can observe the grid growing to fit the data.

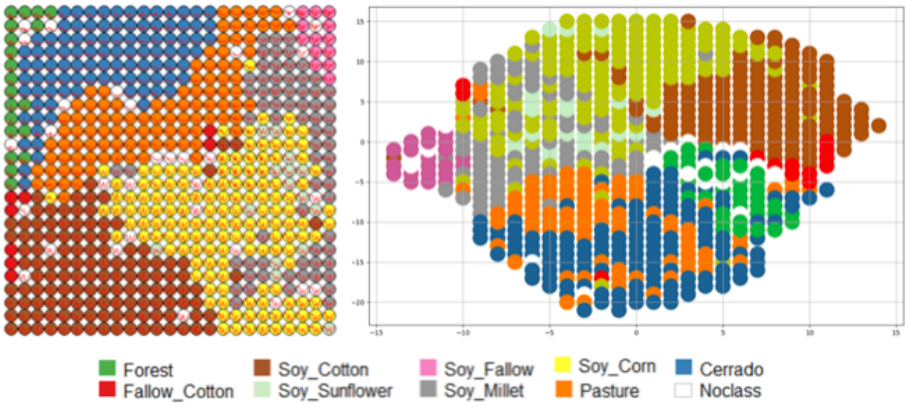

Fig. 5. SOM x GSOM network topology.

4.3 Performance

Trying to approximate SOM classification accuracy, several GSOM running experiments were made, testing different learning rates, spread factors and number of iterations. We noticed that, for the given data set, the GSOM map increased significantly on the number of neurons between iterations 1 and 5. Between iterations 6 and 10, the map had a small growing of the neuron number. After iteration 11, we faced almost no growing. Analyzing the smoothing phase, we noticed that after 5 iterations, the difference between the neuron weights after each iteration were above 0.001. This facts lead us to fix the number of iterations on growing phase in 10, and the number of iterations on smoothing phase in 5.

As the SOM algorithm has a fixed grid, each iteration takes approximately the same time to run. On the GSOM growing phase, the increasing number of neurons implies on variable time spent on each iteration. Figure 6 illustrates the time spend by SOM (with 25 x 25 fixed grid) and GSOM parametrized with 10 Growing iterations and 5 Smoothing iterations. Both algorithms were executed 10 times, using the same learning rate and the same neighborhood update functions.

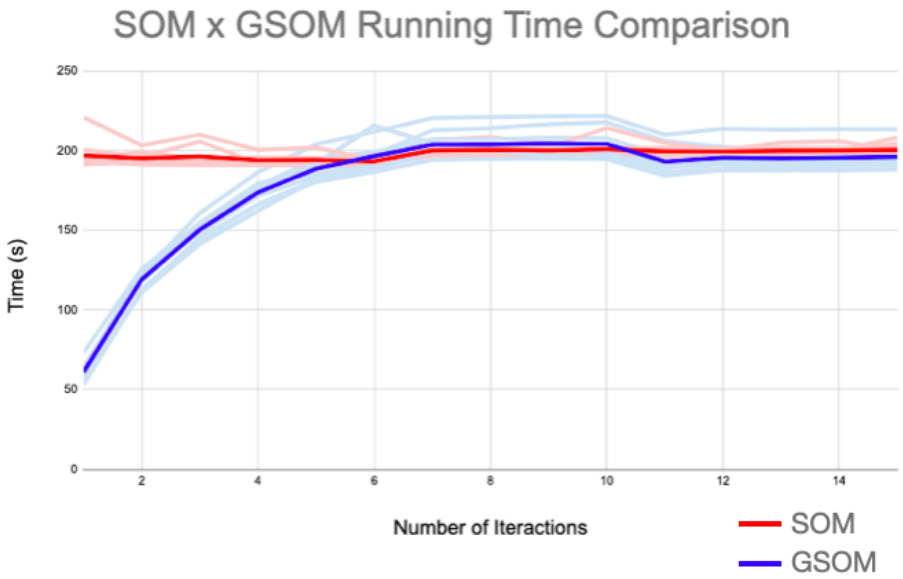

Fig. 6. SOM x GSOM performance comparison.

4.4 Cluster accuracy

After the unsupervised clustering provided by SOM, each neuron is analyzed, and the samples associated to this neuron are counted. All neurons used by SOM are labeled, using the label of the majority samples associated to it. After that, we are now able to check the accuracy of each neuron, and consequently, the whole map accuracy.

In order to compare the cluster accuracy, the same implementation of SOM and GSOM algorithms were executed 10 times, using the same learning rate and the same neighborhood update functions. As the Python implementation slows down the overall solution performance, only 15 iterations of each algorithms were ran. Before that, several GSOM executions were ran in order to generate a grid map with almost the same neurons quantity of a 25 x 25 SOM. The results are showed on Table 1. We noticed that, besides a small classification increment,

GSOM presents significant results of +11.7% and +14.5% on Fallow-Cotton and Soy-Sunflower clustering respectively.

Table 1. 15 SOM iterations x 15 GSOM iterations - Accuracy for each cluster.

Cluster	SOM Accuracy	GSOM Accuracy	GSOM – SOM
Cerrado	96.4	97.6	1.2
Fallow-Cotton	84.1	95.8	11.7
Forest	98.4	98.4	0.0
Pasture	88.9	93.8	4.9
Soy-Corn	86.1	87.2	1.1
Soy-Cotton	91.1	93.9	2.8
Soy-Fallow	99.9	100.0	0.1
Soy-Millet	82.1	85.9	3.8
Soy-Sunflower	70.3	84.8	14.5
Total Accuracy	90.0	92.8	2.8

In the case study provided by Santos et. al. [13], the best classification scenario was obtained with the time series presented to a SOM parameterized with grid size = 25 x 25, learning rate = 1 and number of iteration = 100. The SOM implementation used in this experiments was the Kohonen R package [15]. It provides the original SOM functionality with good performance due to its Rcpp implementation [3]. The use of C++ code mixed with R code provides a significant reduction on the algorithm running time, allowing an increase on the number of iterations without a huge impact on the overall execution time. After 100 iterations, the overall classification accuracy was 93%.

The same time series attributes were concatenated and presented to a GSOM, parameterized with spread factor = 1, learning rate = 1 and number of iteration = 15 (10 iteration for growing and 5 iteration for smoothing). The overall cluster accuracy of GSOM was 93%. The cluster accuracy for each cluster is presented in Table 2. We can notice that, despite of a small classification decrease on Fallow-Cotton and Pasture clusters, GSOM presents a significant increment of the Soy-Sunflower clustering accuracy of +11.7%. Also, the overall sample cluster was pretty similar and stated as 93.0% for both algorithms.

5 GSOM Neighborhood Analysis

A relevant SOM property that needs to be validated during the GSOM tests is the neighborhood topography maintenance. On a regular SOM, similar satellite image time series are grouped into nearness neighborhoods, even if these time series are labeled as different classes. The distribution on the SOM grid over the iterations tends to cluster satellite image time series of a specific sample on the same closer neighborhood. On the specific case of satellite image time series associated to land use and cover ground samples, the neighborhood maintenance in the clustering process is useful to identify sampling outliers.

Table 2. 100 SOM iterations x 15 GSOM iterations - Accuracy for Each Cluster.

Cluster	SOM Accuracy	GSOM Accuracy	GSOM – SOM
Cerrado	97.3	98.2	0.9
Fallow-Cotton	85.7	82.3	–3.4
Forest	99.3	98.5	–0.8
Pasture	97.3	94.5	–2.8
Soy-Corn	84.0	84.4	0.4
Soy-Cotton	95.5	95.2	–0.3
Soy-Fallow	100.0	100.0	0.0
Soy-Millet	90.3	88.9	–1.4
Soy-Sunflower	76.9	88.6	11.7
Total Accuracy	93.0	93.0	0.0

In this case study, we can conclude that GSOM also keeps this property, as shown in Figure 7. The map highlights the nearest neighbors of Neuron 1 (N1). We also point out some random distant neighbors in blue (N593, N607, N610, and N615). Analyzing the distance between the weight vectors of Neuron 1 and the weight vectors of the nearest and some distant random neighbors, we can notice that the distance between the neuron weights grows as we move away from the comparing source. We can visually inspect the time series generated by each of this neurons weights, to check the distance between the nearest neighbors and the distant ones, as shown in Figure 8.

The sum of the difference between Neuron 1 weights of its nearest neighbors is 26.94. As we move away from Neuron 1, this distance grows. As an example, the distance between the selected distant random neurons weights and Neuron 1 weights is 57.15. These distances are summarized on Table 3. This property was analysed for other neurons in the map and similar results were found.

6 Final Remarks

For the given satellite image time series clustering problem, on the given data set (Mato Grosso State, Brazil vegetation ground samples), GSOM seems to be a suitable alternative to SOM. After small customization and adaptation, the

Table 3. Distance between Neuron 1 weights and its nearest/farther neighbors.

N1 - Nearest	Distance	N1 - Farther	Distance
N1 - N0	7.27	N1 - N593	14.80
N1 - N3	5.67	N1 - N607	16.43
N1 - N5	6.28	N1 - N615	13.95
N1 - N4	7.72	N1 - N610	11.97
Neighborhood Distance	26.94	Neighborhood Distance	57.15

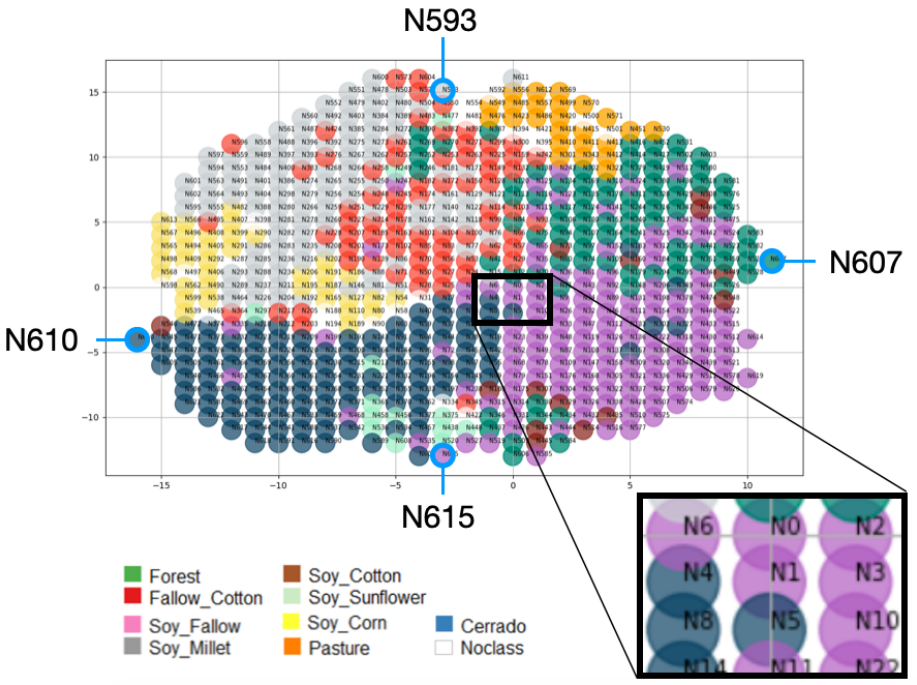

Fig. 7. GSOM grid sample extract - neighborhood analysis.

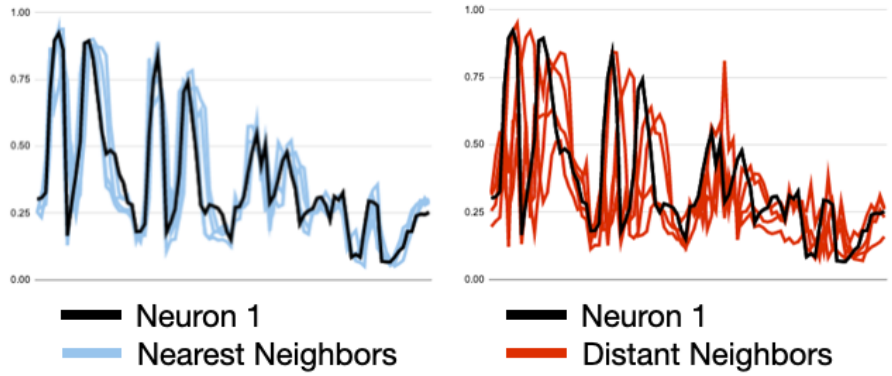

Fig. 8. Neighborhood Analysis - Nearest x Random Distant Neighbors.

GSOM generated map grew as expected, reaching the same amount of neurons of the equivalent SOM, but with a different network topology. GSOM also keeps an important characteristic of SOM, the neurons neighborhood influence.

The cluster accuracy reached by GSOM was similar to the equivalent SOM. It also had a small advantage on the overall clustering time, thanks to its growing

phase, when the neuron number is smaller than SOM. We were also able to test the main GSOM characteristic that initiates this work, which was the capability of clustering the dataset without specifying the grid size. The Spread Factor usage was a simple and effective task, allowing users without previous algorithm knowledge to cluster the data.

The source code used in this work and the dataset files are available on GitHub [7]. The current implementation is a prototype, that will be evolved in order to become a consolidated product in the future. Future works also include applying the GSOM algorithm to other land use and cover time series datasets, in order to confirm the benefits showed on the experiment presented in this work.

References

1. Alahakoon, D., Halgamuge, S. K., and Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. In: IEEE Transactions on Neural Networks, pp. 601—614. (2000).
2. Bagan, H., Wang, Q., Watanabe, M., Yang, Y., Ma, J.: Land Cover Classification From Modis EVI Time-series data using SOM neural network. In: International Journal of Remote Sensing (2005).
3. Eddelbuettel, D. and François, R.: Rcpp: Seamless R and C++ Integration. In: Journal of Statistical Software (2011).
4. Ferreira, K. R., Santos, L., Picoli, M. C. A.: Evaluating distance measures for image time series clustering in land use and cover monitoring. In: Machine Learning for Earth Observation Workshop (2019).
5. Flexer, A.: On the use of self-organizing maps for clustering and visualization. In: Journal Intelligent Data Analysis (2001).
6. Gomez, C. and White, J. C. and Wulder, M. A.: Optical remotely sensed time series data for land cover classification: A review. In: ISPRS Journal of Photogrammetry and Remote Sensing (2016).
7. GSOM Python Implementation Repository, https://github.com/rodrigosales/GSOM. Last accessed 3 May 2020.
8. Kohonen, T. and Schroeder, M. R. and Huang, T. S.: Self-Organizing Maps. Springer-Verlag. 3rd Edition (2001).
9. Natita, W. and Wiboonsak, W. and Dusadee, S.: Appropriate Learning Rate and Neighborhood Function of Self-organizing Map (SOM) for Specific Humidity Pattern Classification over Southern Thailand. In: International Journal of Modeling and Optimization (2016).
10. Pasquarella, V. J. and Holden, C. E. and Kaufman, L. and Woodcock, C. E.: From imagery to ecology: leveraging time series of all available Landsat observations to map and monitor ecosystem state and dynamics. In: Remote Sensing in Ecology and Conservation (2016).
11. Picoli, M., Camara, G., Sanches, I., Simoes, R., Carvalho, A., Maciel, A., Coutinho, A., Esquerdo, J., Antunes, J., Begotti, R., Arvor, D. and Almeida, C.: Big earth observation time series analysis for monitoring Brazilian agriculture. In: ISPRS Journal of Photogrammetry and Remote Sensing (2018).
12. Python Software Foundation. Python Language Reference, version 3.7.2. Available at http://www.python.org.

13. Santos, L. A. and Ferreira, K. R. and Picoli, M. and Camara, G.: Self-Organizing Maps in Earth Observation Data Cubes Analysis. In: International Workshop on Self-Organizing Maps, pp. 70–79 (2019).
14. Vasighi, M. and Abbasi, S.: Multiple growing self-organizing map for data classification. In: International Symposium on Artificial Intelligence and Signal Processing (2017).
15. Wehrens, R. and Buydens, L.: Self and Super-Organizing Maps in R: The kohonen Package. In: Journal of Statistical Software (2007).