International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.5 (2016), pp.15-24
http://dx.doi.org/10.14257/ijsip.2016.9.5.02

Clustering Algorithms in Echo State Networks

Wesam M. Ashour[1]*, Abdallatif S. Abu-Issa[2] and Olaf Hellwich[3]

[1]*Islamic University of Gaza, Palestine*
[2]*Birzeit University, Palestine*
[3]*Technical University of Berlin, Germany*
[1]*washour@iugaza.edu.ps*, [2]*abuissa@birzeit.edu*, [3]*olaf.hellwich@tu-berlin.de*

Abstract

In this work, we develop a new method of setting the input to reservoir and reservoir to reservoir weights in echo state machines. We use a clustering technique which we have previously developed as a pre-processing stage to set the reservoir parameters which at this stage are prototypes. We then use these prototypes as weights in the standard architecture while setting the reservoir to output weights in a standard manner. We show results on a variety of data sets in the literature which show that this method out-performs a standard random echo state machine.

Keywords: *Echo state machines, Time series data, Clustering, Reservoir*

1. Introduction

There are many clustering algorithms but few [16-17] which are specifically designed for time series data. Since such data may contain *e.g.* Periodical such as seasonal data or, particularly for financial time series, be non-stationary, clustering such data provides special challenges. In this paper, we use an echo state network (ESN) in order to cluster time series data.

The standard ESN uses random weights in the reservoir (see below) however in our work, we create structure in these weights by using an existing clustering algorithm to set these parameters and most importantly subsequently use these parameters as feed forward weights in the same way that standard ESNs do.

In the next subsections, we review echo state networks and then existing clustering techniques before describing our new work. We give exemplar clustering results on standard data sets used in the literature.

1.1. Echo State Networks

Echo state networks (ESNs) [9], [21-24] consist of three layers of 'neurons': an input layer which is connected with random and fixed weights W_{in} to the next layer which forms the reservoir. The neurons of the reservoir are connected to other neurons in the reservoir with a fixed, random, sparse matrix of weights W. Typically only about 10% of the weights in the reservoir are non-zero. The reservoir is connected to the output neurons using weights which are trained using error descent. The constructed reservoir is sensitive to the initial weights. Different random weights may cause different capabilities in reservoirs with respect to representing the time series data.

To construct the reservoir, we have the following equation:

$$x(t) = f(W_{in}u(t) + W x(t-1)) \quad (1)$$

ISSN: 2005-4254 IJSIP

where typically $f(.) = \tanh(.)$, t is the time index, $W_{in} \in \mathbb{R}^{N_x \times N_u}$, $W \in \mathbb{R}^{N_x \times N_x}$, N_u is the number of input neurons and N_x is the number of reservoir units. The feed forward stage is given by

$$y = W_{out}\, x \tag{2}$$

where Wout $\in R^{N_x \times N_y}$ and N_y is the number of output neurons. This is followed by a supervised learning of the output weights, W_{out}. W_{out} weights are changed in a training process using gradient descent methods or other classical tools for solving a simple linear regression.

It is often stated that the W weights should be such that the spectral radius (its greatest eigenvalue) is less than 1 to ensure stability in the reservoir when there is no input (see (1)). However a more useful heuristic for the more usual conditions (in which there is a non-zero input) is that there should be a playoff between the magnitude of the reservoir-reservoir weights, W, and those from the inputs, W_{in}: the larger W is, the more memory of previous values can be retained but of course we cannot ignore the inputs entirely. Most research effort has gone into giving the reservoir weights some structure, either by pre-training with, for example, a self-organising map [9-19] or by fixing the topology of the reservoir [21-25]. In this paper, we propose a new method to find an optimal set of weights to construct a more useful reservoir

1.2. Data Clustering

Data clustering techniques are an important aspect used in many fields such as data mining [20], pattern recognition and pattern classification [3], data compression, machine learning [8], image analysis [26], and bioinformatics [22]. The purpose of clustering is to group data points into clusters in which the similar data points are grouped in the same cluster while dissimilar data points are in different clusters. Measures of quality of clustering are based on obtaining high intra-cluster similarity and low inter-cluster similarity.

The K-means algorithm is one of the most frequently used investigatory algorithms in data analysis. The algorithm attempts to locate K prototypes or means throughout a data set in such a way that the K prototypes in some way best represent the data. It is an iterative algorithm in which K means are spread throughout the data and the data samples are allocated to the mean which is closest (often in Euclidean norm) to the sample. Then the K means are repositioned as the average of data points allocated to each mean. This continues until stable convergence is reached. The K-means algorithm is one of the first which a data analyst will use to investigate a new data set because it is algorithmically simple, relatively robust and gives 'good enough' answers over a wide variety of data sets: it will often not be the single best algorithm on any individual data set but it may be close to the optimal over a wide range of data sets. However the algorithm is known to suffer from the defect that the means or prototypes found depend on the initial values given to them at the start of the simulation: a typical program will converge to a local optimum. There are a number of heuristics in the literature which attempt to address this issue but, at heart, the fault lies in the performance function on which K-means is based.[18] proposed a global K-means algorithm, an incremental approach to clustering that adds one cluster prototype at a time through a deterministic global search consisting of N (the data size) executions of the K-means; this algorithm can obtain equivalent or better results than the standard K-means, but it suffers from high computation cost and at the same time gives no guarantee to find the optimum.

Arthur and Vassilvitskii [2] improved the K-means algorithm by substituting the random allocation of the prototypes with a seeding technique. They give

experimental results that show the advantage of this algorithm in time and accuracy.

In [4-7] we derive a family of new clustering algorithms that solve the problem of sensitivity to initial conditions in the K-means algorithm.

This paper is composed as follows. In Section 2, we describe the proposed algorithm. We show how to find an optimal set of weights based on clustering to construct more useful reservoir. We describe the clustering algorithm that we have used. In Section 3 we show our simulation and experimental results. Finally in Section 4, we present the conclusion of our work.

2. The Proposed Method

To construct the reservoir we need two matrices of weights, W_{in} and W. These two matrices are used to map the input space u into the reservoir space x as shown in eq. (1). The advantage of reservoir space is that it captures and represents the input time series data in a useful way by maintaining the dynamic history between samples. In the standard version of constructing the reservoir, random weights have been used. However, random weights may cause inaccurate construction of the reservoir and, almost by definition of random, is liable to be less than optimal.

Some papers [9-19] have proposed methods for selecting the weights values that give better reservoir construction. [19] has introduced a model called Self-Organised Reservoir. In this model, Self-Organising map [15] has been used for fixing the input and reservoir weights prior to training the output weights. Recently, [9] has investigated the idea of using Scale Invariant Maps (SIM) [10- 12] to create the reservoir. However, those methods have a high cost of processing and are thus time consuming. In this paper we propose a novel method that based on clustering for finding an optimal set of weights for both W_{in} and W. This new method, through finding a better set of weights, enhances mapping from input to reservoir space and constructs a more useful reservoir. In the proposed algorithm, we have used a previously developed clustering algorithm, IWK, [5-7] to find the weights for both W_{in} and W.

2.1. Inverse Weighted K-means Algorithm IWK

Consider the performance function

$$\sum_{i=1}^{N}\left[\sum_{j=1}^{K}\frac{1}{\left\|x_i - m_j\right\|^P} * \min_k\left(\left\|x_i - m_k\right\|^n\right)\right] \tag{3}$$

where, K is the number of clusters, N is the number of data points, and n and p are values of the exponent.

In this objective function, we multiply two functions together. One of them is the minimum function which calculates the distance between each data point and its closest prototype (sometimes also called the seed or centroid). This function is important to help in clustering data, but it has a limitation which causes dead prototypes and also convergence to a local optimum. The problem for this function is that each prototype responds only to data points that are closest to this prototype and is not affected by other data points. Thus it is sensitive to the prototypes' initialisation. This limitation has been overcome by multiplying the minimum function by the inverse weighted function which gives a relationship between all data points and all prototypes. The purpose of the inverse weighted function is to provide a good learning process for all prototypes, without losing the advantage of the minimum function which helps to find the clusters. This function makes the prototypes, in the derived learning process,

respond to all the data points not only to the closest data points to the prototypes. Thus each prototype before being moved to any new location, responds to all the other prototypes' positions and, in particular, to their relative locations with respect to the data points, and hence it is possible to identify the free clusters that are not recognized by the other prototypes [5-7].

To create a learning rule and find the new locations of the prototypes that give iteratively better performance, we need to find the partial derivative of the performance function with respect to m_{k*}, which is the closest prototype to x_i, and with respect to m_j which represent the prototypes that are not the closest to x_i. Solving this over all the data set results in

$$m_r(t+1) = \frac{\sum_{i \in V_r} x_i a_{ir} + \sum_{i \in V_j, j \neq r} x_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \tag{4}$$

where V_r contains the indices of data points that are closest to m_r, V_j contains the indices of all the other points and

$$a_{ik} = -(n-p)\|x_i - m_r(t)\|^{n-p-2} - n\|x_i - m_r(t)\|^{n-2} * \left(\sum_{j \neq k^*} \|x_i - m_j\|^{-p} \right) \tag{5}$$

$$b_{ir} = p * \frac{\|x_i - m_{k^*}\|^n}{\|x_i - m_r(t)\|^{p+2}} \tag{6}$$

A more complete discussion of this and other methods can be found in [5-7].

2.2. Supervised Learning

After finding an optimal set of W_{in} and W weights to construct a more useful reservoir, we use a linear regression supervised method to train the output weights which generate the output vectors that match the desired output vectors.

$$y = W_{out}\, x \tag{7}$$

where: y is the output vector, W_{out} is the trained weights to generate the desired output, and x is the reservoir.

2.3. The Algorithm

In this section, we describe the main steps of the proposed method in Algorithm 1. It is important to recognize that in stage 8, we are using the newly constructed input

Algorithm 1 Description of the Proposed Algorithm

INPUT: $\{u(t), y_{target}(t) : t = 1, \ldots, T\}$, $W_{in} \in R^{Nx \times Nu}$, $W \in R^{Nx \times Nx}$, $W_{out} \in R^{Nx \times Ny}$, N_x

OUTPUT: MSE, W_{in}, W.

1: Initialise randomly W_{in}, W and W_{out}.
2: Use W_{in} as initial prototypes and apply IWK clustering algorithm to the input data set.
3: After convergence in step 2, assign the resulted prototypes to W_{in}.
4: Construct the reservoir using eq.(1).
5: Use W as initial prototypes and apply the IWK clustering algorithm to the reservoir.
6: After convergence in step 5, assign the resulted prototypes to W.
7: Construct the reservoir again, but this time with the calculated set of weights W_{in} and W.
8: Apply linear regression supervised learning algorithm to train the output weights W_{out}.

9: After training the output weights in step 8 and convergence, calculate the outputs using eq. (7).
10: Finally, calculate the Mean Square Error MSE using:

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y(t) - y_{target}(t))^2 \quad (8)$$

to reservoir weights, W_{in} and reservoir to reservoir weights, W in the standard manner *i.e.*, exactly as in 1. Thus the parameters which were previously cluster prototypes or centres are now multiplicative parameters. This is very different from those methods which instantiated the reservoir with SOM or SIM [9-19] which continued to use the prototypes in the standard manner. Note that at stage 8, we will have maximum output values for inputs which have greatest projection on the input weights and for which simultaneously, the current reservoir activations have greatest projection on the reservoir weights which contain historical information of the previous inputs. It is this which gives power to our method.

3. Simulations

To test and compare the resulting reservoirs constructed with random weights and with the proposed algorithm, we have the following main steps:
1. Construct the reservoir using random weights for W_{in} and W.
2. Apply linear regression supervised learning algorithm to train the output weights W_{out}, and then find the outputs y using eq. (7).
3. Using eq. (8), calculate the mean square error (MSE).
4. Repeat the run many times (specified below) and record each time the calculated (MSE).
5. Repeat all the previous steps, but this time with the reservoir that has been constructed with the proposed algorithm.
6. Finally, compare the results.

Experiment 1: Freedman's non linear time series dataset [14-21]:
$y_{target}(t+1) = f(y_{target}(t))$, where:

$$\begin{cases} 2x & if\ x \le 0.5 \\ 2 - 2x & if\ x > 0.5 \end{cases} \quad (9)$$

In this experiment, we have used the first 60 samples of Freedman time series dataset, $y_{target}(0) = 0.001$. We have constructed the reservoir twice, once with random weights and the second time with the proposed algorithm. Then we have calculated the MSE after applying the supervised algorithm which is required to find the output weights W_{out}. We have run the code 10 times (reservoir units = 10) and recorded the MSE results for each run in Table 1. In Table 1, the average MSE we got with using random weights, 0.4411, is larger than the average MSE we got with using the proposed algorithm, 0.0017. The proposed algorithm constructs a more useful reservoir which represents the data with its history.

Table 1. Mean Square Error (MSE) for 10 Times Run, Freedman Data. Results with using Random Weights and with using the Proposed Algorithm. Reservoir Units = 10

	Random Weights 10 units	Proposed Algorithm 10 units
1	0.1748	0.0016
2	0.2159	0.0012
3	0.0071	0.0012
4	0.0098	0.0026
5	2.5484	0.0012
6	0.1930	0.0015
7	0.1171	0.0020
8	0.8503	0.0011
9	0.0139	0.0011
10	0.2804	0.0031
Average	0.4411	0.0017

Experiment 2: Fixed 10th order Narma time series dataset [1]:

$$y_{target}(t+1) = 0.3y_{target}(t) + 0.05y_{target}(t)\sum_{k=0}^{9} y_{target}(t-k) + 1.5s(t-9)s(t) + 0.1 \quad (10)$$

where: $s(t) \in Unif[0.0,0.5]$.

In this experiment, we have omitted the first 50 samples and used the next 300 samples of Narma 10th order time series dataset. The first 50 samples may be considered burn-in data allowing the network to settle to its standard regime. Similar to the previous experiment we run the code 10 times and record the results in Table 2. As shown in Table 2, the proposed algorithm gives better results than that given by using the random weights.

Table 2. Mean Square Error (MSE) for 10 Times Run, Narma 10 Data. Two Columns of Results, One for Random Weights, and One for the Proposed Algorithm. Reservoir Units = 10

	Random Weights 10 units	Proposed Algorithm 10 units
1	0.1144	0.0069
2	0.0655	0.0067
3	3.3000	0.0067
4	0.4032	0.0068
5	9.7462	0.0070
6	1.2837	0.0069
7	0.8542	0.0069
8	0.0140	0.0068
9	0.1007	0.0069
10	0.1795	0.0068
Average	1.6061	0.0068

Experiment 3: Henon map time series dataset [13]:

$$y_{target}(t+1) = 1 - 1.4y_{target}(t)^2 + 0.3y_{target}(t-1). \quad (11)$$

In this experiment, we have used the first 1000 samples of the Henon map, initially, $y_{target}(t-1) = 0, y_{target}(t) = 0.1$ and then iterate. For reservoir units, we have used 30, 60 and 100 units.

Table 3. Mean Square Error (MSE) for 100 Times Run, Henon Data. Results of 30, 60 and 100 Reservoir Units with Random Weights and with the Proposed Algorithm. Average2 (90 runs) is the Average after Removing the Worst 10 MSE Results. Average3 (80 runs) is the Average after Removing the Worst 20 MSE Results

	Random Weights 30 units	Proposed Algorithm 30 units	Random Weights 60 units	Proposed Algorithm 60 units	Random Weights 100 units	Proposed Algorithm 100 units
1	0.0971	0.0013	0.3678	0.0011	0.5468	0.0010
2	0.0804	0.0019	0.1698	0.0010	0.5488	0.0004
3	0.2027	0.0033	0.2447	0.0008	0.9744	0.0008
4	0.2530	0.0020	1.8373	0.0008	0.4549	0.0003
5	0.0064	0.0013	0.1867	0.0005	0.1924	0.0003
6	0.0148	0.0027	0.4786	0.0006	0.5488	0.0005
7	0.2950	0.0013	0.4869	0.0020	0.3340	0.0002
8	0.0643	0.0022	0.0774	0.0005	0.5744	0.0002
9	0.0216	0.0019	0.3943	0.0005	0.3722	0.0002
10	0.0494	0.0014	0.3718	0.0004	0.1017	0.0002
...	...	...	...	...	...	...
100	0.0750	0.0034	0.0368	0.00083	0.3974	0.00051
Average1	0.1334	0.0026	0.5507	0.000739	1.747	0.000556
Average2	0.0824	0.0023	0.2948	0.000644	0.5365	0.000458
Average3	0.0604	0.0022	0.2205	0.000589	0.4335	0.000401

In Table 3, first 10 rows show the MSE result after running the code 10 times with reservoir units 30, 60 and 100 respectively. The last 3 rows show the average of MSE results. Average1 is the average MSE for 100 runs. Average2 (90 runs) is the average after removing the worst 10 MSE results. Average3 (80 runs) is the average after removing the worst 20 MSE results. We have two columns for each reservoir units number. he first column shows the MSE resulted from using random weights, while the second column shows the MSE resulted from using the proposed algorithm. We have plotted the MSE results for 100 runs with reservoir units 30, 60 and 100 in Figures 1, 2 and 3, respectively. In each figure, the dotted line represents the average MSE for 100 runs. The left figures show the results with using random weights, while the right figures show the results with using the proposed algorithm. As shown in Table 3 and Figures 1, 2 and 3, we can see that the proposed algorithm provides superior results comparable to random weights (note the vertical axes scales). It is also worth noting

that the random algorithm gives several examples of really poor initialisation of W_{in} and W as shown in the large variation in output errors.

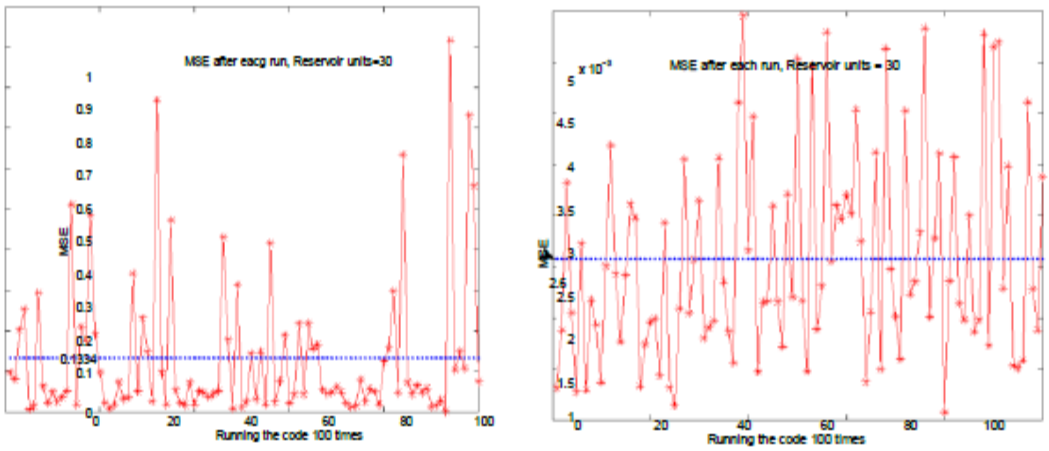

Figure 1. 30 Reservoir Units. Left: MSE error for 100 Runs with random Weights. Right: MSE Error for 100 Runs with Proposed Algorithm

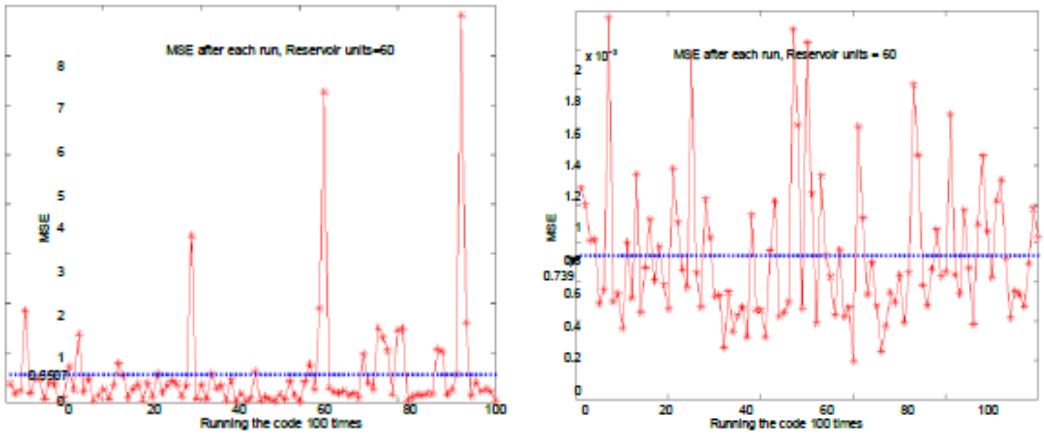

Figure 2. 60 Reservoir Units. Left: MSE Error for 100 Runs with Random Weights. Right: MSE Error for 100 Runs with Proposed Algorithm

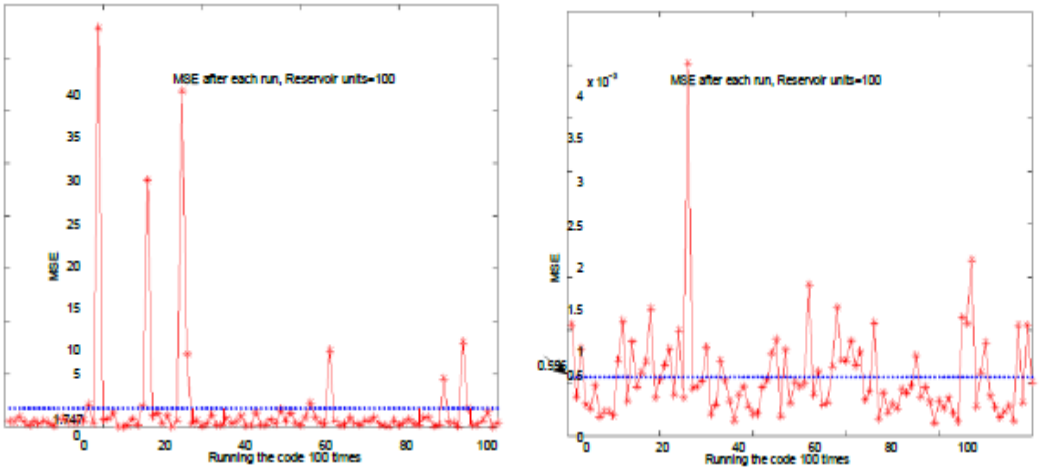

Figure 3. 100 Reservoir Units. Left: MSE Error for 100 Runs with Random Weights. Right: MSE Error for 100 Runs with Proposed Algorithm

4. Conclusion and Future Work

We have developed a new method for constructing non-random reservoirs for echo state networks and shown that the new method out-performs the standard random networks on
3 standard data sets: it gives a lower mean squared error and also much less variance than randomly constructed reservoirs do.

The most important feature of our method is the dual nature of the parameters: during the first stage of the process, W_{in} and W both act as prototypes and thus are useful in identifying different regimes in the time series. During the second stage, when the W_{out} weights are being found, W and W_{in} are fixed and are used as multiplicative parameters. Thus at this stage, the reservoir activation is function of the projection of the input data and its current memory on the prototypes from the first stage and thus the reservoir neurons which are most active are those which are most aligned with the current input and history; the parameters are now acting as a measure of how like the smoothed historical data the current data is.

Future work will perform a comparison between the clustering methods in this paper with other clustering techniques in the context of echo state machines as predictors of the future.

Acknowledgments

The first author would like to extend his sincerest thanks and appreciation to the Deutscher Akademischer Austauschdienst (DAAD) for their support and grant to accomplish and complete this research.

References

[1] F. Atiya and A. G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence", In IEEE Transactions on Neural Networks, vol. 11, **(2000)**, pp. 697–709.
[2] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding", In The eighteenth annual ACM-SIAM symposium on Discrete algorithms", **(2007)**, pp. 1027-1035.
[3] D. Balya, "Cnn universal machine as classification platform: An art-like clustering algorithm", International Journal of Neural Systems, vol. 13, no. 6, **(2003)**, pp. 415–425.
[4] W. Barbakh, M. Crowe and C. Fyfe, "A family of novel clustering algorithms", In 7th international conference on intelligent data engineering and automated learning", IDEAL2006, Springer. ISSN 0302-9743, **(2006)**, pp. 283-290
[5] W. Barbakh and C. Fyfe, "Local vs global interactions in clustering algorithms: Advances over k-means", International Journal of Knowledge-based and Intelligent Engineering Systems, ISSN 1327-2314, vol. 12, no. 2, **(2008)**, pp. 83–99.
[6] W. Barbakh and C. Fyfe, "Online clustering algorithms", International Journal of Neural Systems (IJNS), ISSN 1327-2314, vol. 18, no. 3, **(2008)**, pp. 1–10.
[7] W. Barbakh, Y. Wu and C. Fyfe, "Non-standard exploratory data analysis", Springer, **(2009)**.
[8] M. Celebi, "Effective Initialization of K-means for Color Quantization", In Proc. of the IEEE International Conference on Image Processing, DOI: 10.1.1.151.5281,**(2009)**, pp.1649–1652.
[9] S. Basterrech, C. Fyfe and G. Rubino, "Initializing echo state networks with topographic maps", In 2nd International Conference on Morphological Computation, ICMC, **(2011)**.
[10] C. Fyfe, "A scale invariant feature map", Network: Computation in Neural Systems, vol. 7, **(1996)**, pp. 269–275.
[11] C. Fyfe, "Hebbian Learning and Negative Feedback Artificial Neural Networks", Springer, **(2004)**.
[12] C. Fyfe, "Two topographic maps for data visualization", Data Mining and Knowledge Discovery, ISSN 1384-5810, vol. 14, **(2007)**, pp. 207–224.
[13] M. Henon, "A two-dimensional mapping with a strange attractor", Comm. Math. Phys., 50:69–77, 1976.
[14] R. Hyndman, "Time series data library", Available: http://robjhyndman.com/TSDL.
[15] T. Kohonen," Self-Organizing Maps", volume 30. Springer Series in Information Sciences, third edition, **(2001)**.
[16] L. Li and B. A. Prakash, "Time Series Clustering: Complex is Simpler", In Proceedings of the 28th International Conference on Machine Learning, Bellevue, **(2011)**.
[17] T.W. Liao, "Clustering of time series data - a survey", Pattern Recognition, vol. 38, no.11, **(2005)**, pp.1857–1874.

[18] A. Likas, N. Vlassis and J. J. Verbeek. ”The global k-means clustering algorithm”, Pattern Recognition, no. 36, **(2003)**, pp. 451–461.
[19] M.Lukoˇseviˇcius. ”On self-organizing reservoirs and their hierarchies”, Technical Report 25, Jacobs University, Bremen, **(2010)**.
[20] G. B.-Orgaz, H. D. Menendez and D. Camacho, ”Adaptive K-means algorithm for overlapped graph clustering”, International Journal of Neural Systems, 1250018, vol. 22, no. 5,**(2012)**, pp.1–19.
[21] A. Rodan and P. Tinˇo, “Minimum complexity echo state network”, IEEE Transactions on Neural Networks, vol. 22, **(2011)**, pp. 131–44
[22] L. Wang, M. Jiang, Y. Lu, M. Sun and F. Noe. ”A Comparative study of clustering methods for molecular data”, International Journal of Neural Systems, vol. 17, no. 06, **(2007)**, pp.447–458.
[23] T. D. Wang, X. Wu and C. Fyfe, ”Comparative study of visualisation methods for temporal data”, 2012 IEEE Congress on Evolutionary Computation, **(2012)**.
[24] T. D. Wang and C. Fyfe. ”The role of structure size and sparsity in echo state networks for visualisation”, The United Kingdom Conference on Computational Intelligence, **(2011)**.
[25] Y. Xue, L. Yang and S. Haykin, ”Decoupled echo state networks with lateral inhibition”, Neural Networks, vol. 3, **(2007)**, pp. 365–376.
[26] M. Al- Zoubi, A. Hudaib, A. Huneiti and B. Hammo, ”New Efficient Strategy to Accelerate k-Means Clustering Algorithm”, American Journal of Applied Science, vol. 5, no. 9, **(2008)**, pp.1247-1250.

