

## CLUSTERING TIME SERIES OF DIFFERENT LENGTH USING SELF-ORGANISING MAPS

S. Parshutin

*Keywords: clustering, time series, self-organising maps, neural networks, time warping*

### 1. Introduction

Each year technologies within the scope of the artificial intelligence become more popular and find new applications in many new fields, such as medicine, militaries, security business and economics. One of the attractive fields for the application of the artificial intelligence is the monitoring of the demand curve of the product. Modern computer and information technologies make it possible for companies to gather huge amounts of data – amounts that a human cannot handle by bare hands. This is one of the main reasons for companies to apply systems for automatic analysis of incoming data on products.

For similarity search between demand curves of the products, clustering of the time series most often is used. A various number of clustering methods exist. This work examines a Self-Organising Map based time series clustering method. The main purpose for such a research is a product demand curve analysis task, with a posterior possibility of pointing out the moment when a product changes its life cycle phase from introduction to maturity. The current work is focused on analysing the possibility of using a single self-organising map for analysis of the data, represented by time series of different length.

The main results of this research are the modifications for the SOM-based clustering algorithm, enabling the self-organising map to cluster the time series of different length, as well as the main algorithm for system operation.

### 2. Theory

As it was mentioned earlier, this research considers the self-organising map based time series clustering method. Several models of the self-organising maps exist; two models - Willshaw-von der Malsburg model and a model, offered by Teuvo Kohonen [1, 2] are considered as base ones. To analyse the possibility of self-organising maps to cluster the time series of different length, the Kohonen's model was chosen. The chosen model is described in Section 2.1. Modifications to the model and the main algorithm are presented in Sections 2.2 and 2.3 respectively.

#### 2.1. Kohonen's self-organising map

The Kohonen's map (see Figure 1) is one of the Vector quantization models [1, 2]. Input data is represented as a vector, which is being sent as an input signal to all the neurons of the network. The length of an input vector is equal to the number of weights each neuron has. During the organization process a topological map of an input data is made.

The map organization algorithm begins with the initialization of the map by assigning the randomly chosen small values to the synaptic weights of neurons [1, 2]. Such an

initialization gives a known certainty that in the beginning the map is not organised. The map initialization process is followed by these three main processes [1, 2]:

1. Competition. For each input vector the closeness to each neuron in the map is calculated. The closest neuron is called a winning neuron or a best matching neuron.
2. Cooperation. The best matching neuron defines the position for the centre of the topological neighbourhood of the neurons (see Figure 1) and excites the topologically closest neurons. Let us define the lateral distance between the winning neuron ( $i$ ) and an excited neuron ( $j$ ) as  $d_{i,j}$ , a topological neighbourhood as  $h_{i,j}$ . Neighbourhood function  $h_{i,j}$  is symmetrical to the winning neuron and decreases while distance  $d_{i,j}$  increases. Typical example of  $h_{i,j}$  could be the Gaussian function (1).

$$h_{i(x),j} = \exp\left(-\frac{d_{i,j}^2}{2 \cdot \sigma^2}\right), \quad (1)$$

where  $\sigma$  decreases with time (iterations), most often - exponentially.

3. Synaptic adaptation. This mechanism grants excited neurons a possibility to increase the values of the discriminant functions proportionally to the incoming data by updating the synaptic weights. As a result, the response strength of the winning neuron to the next occurrences of similar data increases.

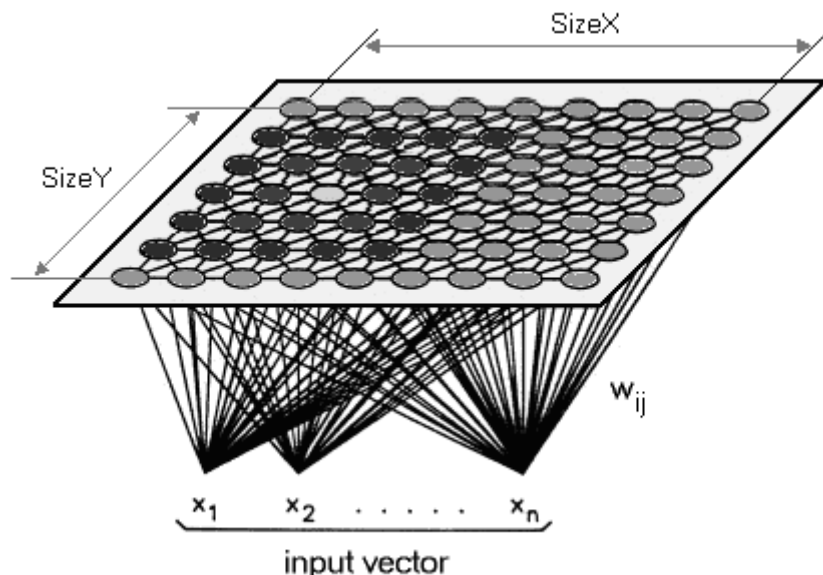


Figure 1. Kohonen's network

## 2.2. Modifications of the SOM

As it was described, the classical Kohonen's model presumes that the length of the input vector and the number of weights of a neuron are equal.

One of the main modifications to be made is the modification of the distance calculation techniques between the input vector and the vector of weights of the neuron. In place of the

Euclidean distance (2), time warping techniques should be used. Such a modification makes it possible to compare data vectors of different length.

$$d_E(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^n (\xi_i + \eta_i)^2}, \quad (2)$$

The time warping algorithm consists of several steps [3, 4]. First, a distance matrix  $n \times m$  is built, where  $n$  and  $m$  are the lengths of the compared data vectors. Each cell of a distance matrix holds a distance  $d_{ij}$  between corresponding points of the vectors compared. Total distance  $D$  between two data vectors is the minimal length of a path – the sum of distances at each step, beginning with the cell  $[1,1]$  and ending with  $[n,m]$ . A movement (steps) in the distance matrix obeys the defined rules.

Most often the  $d_{ij}$  is calculated as an Euclidean distance (2); this can lead to the situation when a single point on one time series maps onto a large subsection of another time series [3]. This is one of the main points why the shortest path searching process must be controlled and, if necessary, corrected with appropriate rules. As a variant – movement rules in the distance matrix. One of possible examples is also described in [3].

### 2.3. Main algorithm

Summing all the above, to solve the task described in Section 1, the following algorithm is presented:

#### 1. Data pre-processing

- 1.1. A proper data format must be chosen for the representation of the data. The data format is very important, as it directly influences the effectiveness of the neural net [5].
- 1.2. Data should be normalised. While working with the real data, most often the real bounds of a parameter are unknown, that is why the *Z-score* normalization is recommended. A standard deviation (3) or a mean absolute deviation (4) can be used.

$$a'_i = \frac{a_i - \bar{a}}{\sigma_a}, \quad (3)$$

$$a'_i = \frac{a_i - \bar{a}}{s_a}, \quad (4)$$

#### 2. Definition of the number of maps and the values for parameters of the map

- 2.1. By using the available information on dataset, such as the length of the records in the dataset, the length distribution of records and a meaning of data, it is necessary to define the number of maps that will process the supplied dataset. This step is necessary because during the dynamic transformation of data the information the data carries is changed. For the data with length of hundreds of points (medical data, series of sensor data, etc) the distance in 5-10 points may not be so critical. But for the data connected with the described task (section 1), and having around 15 points, the difference in one point may already be critical. That is why while analysing the length distribution of the data, it is necessary to define the subsets of records, according to

the length and meaning of the data. The number of defined subsets may be used as the number of self-organising maps in the system, where each SOM processes the records of the defined length interval.

- 2.2. When part 2.1 is finished, the values of the parameters of each map must be set – map parameters, learning parameter, neighbourhood, etc. [1, 2]. It is recommended to use the median of the length distribution of records in a subset, while setting the number of weights. It will allow lessening data deformation during the map organisation process.

### 3. Learning and testing the system

- 3.1. During the system learning process it is necessary to simulate the online data flowing into the system. That is the realisation of the following steps is presumed: presume that the minimal length of a data vector a system can handle is 4 values, then when a time series with 10 values with a life cycle phase switching in period 9 comes, it is necessary to start with sending first 4 values of a time series, then first 5 and so on till 10, each time giving the system the information that the life cycle phase switching period is 9. By learning the system this way the system learns to recognise the life cycle phase switching in the remote period, by having only first several values.
- 3.2. Solving a task like the one defined in Section 1, it is important to measure not only the numerical precision of a system, but also the logical precision. To measure the numerical precision, a *Mean Absolute Error (MAE)* can be used (5).

$$MAE = \frac{\sum_{i=1}^h |A_i - F_i|}{h}, \quad (5)$$

where  $h$  – a total number of the records in the test data set;  $A$  – a period with a life cycle phase switching;  $F$  – period with life cycle phase switching, given by the system. For calculation of a logical precision of a system, the following approach is proposed: an error in the logic of a system is an event that is caused by one of the next two conditions – “There was a life cycle phase switching, but the system says that it was not” and “There was no life cycle phase switching, but the system says that it was”. The number of errors is counted and divided by  $h$ , giving the logical precision of a system for the current test dataset.

## 3. Experimental part

The current section displays an example of a practical realization of the algorithm described and analyses the obtained experimental results.

For building the system, the *Microsoft Excel 2003* was chosen; the programming language is *Visual Basic for Applications*.

### 3.1. Data and description of experiments

For learning the system and testing it, a dataset containing 199 time series with length from 4 through 24 was used. The dataset was supplied in the framework of the ECLIPS project (see the Acknowledgements). The records contain a demand data of a product till the life cycle phase switching period including the switching period and one record after it. This makes the system to work as a reactive one. An example of the data is displayed in Figure 2.

ID	Switching	T1	T2	T3	T4	T5
851	19	-1.04745	-1.13815	-1.05794	-1.01822	-1.07795
420	12	-1.98925	-2.38041	-1.71671	-1.39448	-1.3268
810	7	-4.13698	-1.81955	-0.57268	-0.10214	0.161654
560	5	-3.73487	-2.35698	-0.85928	0.460119	0.178813
240	23	-0.99826	-0.98395	-1.08076	-0.76167	-1.02177
233	3	-1.56203	-0.55239	1.504353	0.281743	

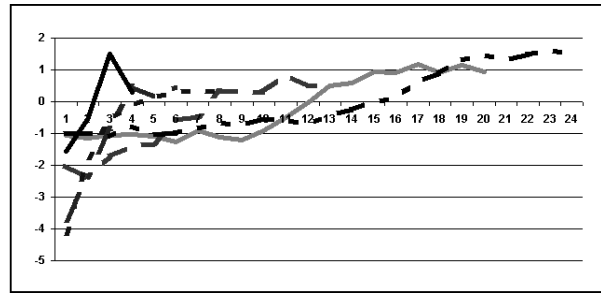


Figure 2. Example of the data (first 5 periods in the table)

The negative values in the data table (see Figure 2) are the result of the data normalization. The *Z-score* normalization method with standard deviation was applied (see Subsection 2.3).

The task of the experiments is to build and teach the system by using the simulation of the online data flowing (see Subsection 2.3), followed by the analysis of the reaction of the system to a new (test) data, also supplied in the online data flowing simulation state.

Each Self-Organising Map is able to process the data of three lengths in a row, that is 4-6, 7-9, 10-12 and so on. By using this condition, the system automatically calculates the number of SOMs necessary to process the supplied dataset. For each SOM in the system the equal values for parameters were chosen, they are displayed in Table 1; only the number of weights of a neuron differs from map to map. For testing the system the 3-fold cross-validation was used.

Table 1.

SOM parameters

SOM		Learning parameter			SIGMA for Gaussian neighbourhood		
Neurons	Size	Start	End	Function	Start	End	Function
25	5x5	0.9	0.01	Exponential	0.5	0.01	Exponential

A detailed description of the functions mentioned in the Table 1 can be found in [1] and [2].

3.2. Results

The system testing results are shown in Table 2.

Table 2.

3-fold cross-validation results

	MAE	Decision	
		Wrong	Correct
Online	2.160893	12.08%	87.92%
Offline	1.450173	50.22%	49.78%

The results in the “offline” row of Table 2 are obtained by testing the system without using the simulation of online data flowing – the whole test record was sent into the system and the response was logged. In other words, the “Offline” state error statistics is obtained by measuring an error at the moment when a test record contains all information before the switching period, at the switching period and at the period after switching.

As can be seen from the experimental results, there is a sort of logic in the decisions made by the system. Almost in 88% of all cases the system was able to take the logically correct decision. But analysing the results obtained when the whole test record was sent to the system (row “offline” of Table 2), in more than 50% of all cases the system was not able to take the logically correct decision. Nevertheless the mean absolute error is around 2 periods, which may indicate that the system works with a certain precision.

#### 4. Conclusions and future work

The experimental results make it possible to conclude that it is not only theoretically possible to cluster the time series of different length using the Self-Organising Maps; it is also possible in practice. Using one Self-Organising Map for clustering time series of different length reduces the number of calculations and the necessary resources. The received experimental results indicate that a system built in the Microsoft Excel 2003, using the Visual Basic for Applications language, works with a certain precision and is able to give a plausible result.

There are several main targets for future work, such as improvement of the precision of the system by tuning the decision making algorithms used in the system; designing the algorithms that will automatically set the parameters for the Self-Organising Maps, by using the available information about the supplied data; and also the modification of computing techniques that will help to lower the time used for system learning process.

#### Acknowledgements

The present research was partly supported by the ECLIPS project of the European Commission "Extended collaborative integrated life cycle supply chain planning system".

#### References

1. Kohonen T. Self-Organising Maps // Springer, 3<sup>rd</sup> edition, 2001. ISSN:0720-678X, ISBN 3-540-67921-9.
2. Haykin S. Neural Networks. A Comprehensive Foundation. Second edition // Prentice Hall, Russian edition, 2006. ISBN 5-8459-0890-6 (rus), ISBN 0-13-273350-1 (eng).
3. Keogh E. and Pazzani M. Derivative Dynamic Time Warping. 2001. // <http://www.cs.ucr.edu/~eamonn/sdm01.pdf>. Last viewed August 2007.
4. Keogh E. Exact Indexing of Dynamic Time Warping // Proceedings of the 28th International Conference on Very Large Data Bases. August 2002.
5. Parshutin S. Clustering Time Series Using Self-Organising Maps. (Russian edition)// Proceedings of IMSCAI-IV Conference. May 28-30, 2007. ISBN 978-5-9221-0842-3, 465-472.

**Serge Parshutin**, Mg.sc.ing., Ph.D. student, Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: serge.parshutin@cs.rtu.lv

**Paršutins Sergejs. Atšķirīga garuma laika rindu klasterizācija pielietojot pašorganizējošus neironu tīklus**

Šis darbs ir veltīts laika rindu analīzes problēmai. Par svarīgo uzdevumu, saistīto ar laika rindām, var nosaukt datu kopas sadalīšanu atsevišķās grupās – klasteros, ar tālāko mērķi pareģot laika rindu uzvedību nākotnē. Eksistē vairākas klasterizācijas metodes; pētījumi darbā ir saistīti ar laika rindu klasterizāciju, pielietojot pašorganizējošus neironu tīklus. Klasiskie pašorganizējošie neironu tīkli prasa, lai ieejas vektoriem būtu vienāds garums, bet vairākuma gadījumos reālie dati neapmierina tādas prasības. Pētījumi darbā fokusējas uz pašorganizējošo neironu tīklu modernizāciju ar nolūku sasniegt iespēju ar to palīdzību klasterizēt atšķirīga garuma laika rindas. Algoritms, kas ļauj izpildīt aprakstītas darbības, ir izstrādāts un aprakstīts darbā, kā arī ir realizēts praktiski, kam ir izstrādāta specializēta programmatūra, kas apvieno sevī datu transformēšanas un klasterizācijas algoritmus, kā arī iegūto praktisko rezultātu analīzes procedūras. Galvenais darba rezultāts ir algoritms, kas ļauj veikt atšķirīga garuma laika rindu klasterizāciju, pielietojot pašorganizējošus neironu tīklus, kā arī algoritma praktiskais pielietojums un sasniegto rezultātu analīze.

**Parshutin Serge. Clustering time series of different length using Self-Organising maps**

The current work is devoted to the problem of time series analysis. One of the relevant tasks connected with time series is splitting the set of objects into individual groups – clusters for further forecasting the behaviour of time series. A various number of clustering methods exist; this work is focused on the time series clustering method which uses self-organising maps. Classical self-organising maps presume that the input vectors are of the same length, but in the most cases the real data does not satisfy this assumption. The work analyses the problems connected with self-organising maps modification aimed to enable clustering time series of various length. The algorithm that allows accomplishing the described tasks, is not only developed and presented in this work, but is also practically realized. A specialised software solution combining data transformation and clustering algorithms and also practical data analysis procedures are developed. The main result of this work is an algorithm, which allows using self-organising maps for clustering time series of various length as well as the practical use of the algorithm and analysis of the obtained results.

**Паршутин Сергей. Кластеризация временных рядов разной длины с применением карт самоорганизации**

Данная работа посвящена задаче анализа временных рядов. Одной из существенных задач, связанных с временными рядами, является разделение множества данных на группы – кластеры, с целью дальнейшего прогнозирования поведения временного ряда. Существует множество различных методов кластеризации; в данной работе исследуется кластеризация временных рядов с применением карт самоорганизации. Классические карты самоорганизации предполагают, что входящие векторы данных имеют равную длину, однако в большинстве случаев реальные данные не удовлетворяют данному условию. Работа сфокусирована на модификации карт самоорганизации с целью получения возможности кластеризовать данные различной длины. Алгоритм, позволяющий выполнить подобные действия, не только разработан и представлен в данной статье, но и реализован практически, для чего создано специализированное программное обеспечение, реализующее в себе алгоритмы преобразования и кластеризации данных, а также - анализа полученных результатов. Главным результатом данной работы является алгоритм, позволяющий кластеризовать временные ряды различной длины с применением карт самоорганизации, а также - практическое применение этого алгоритма для решения реальной задачи – прогнозирования смены фаз жизненного цикла продукта и анализа полученных результатов.