ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2008-7493

INFORMATION TECHNOLOGY AND MANAGEMENT SCIENCE
INFORMĀCIJAS TEHNOLOĢIJA UN VADĪBAS ZINĀTNE

TIME SERIES ANALYSIS WITH MODULAR NEURAL NETWORKS

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.

Ludmila Aleksejeva, Dr.sc.ing., Assoc.prof., Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658 Latvia, e-mail: laleks@cs.rtu.lv.

Arkady Borisov, Dr.habil.sc.comp., Professor, Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658 Latvia, e-mail: aborisov@cs.rtu.lv.

Keywords: time series analysis, modular neural networks, self-organising maps

1. Introduction

The number of publications and books on neural networks and it applications published in last years speaks in favour that a significantly long time neural networks appear to be one of the popular technologies in domain of forecasting different processes, including non-linear ones. A special place in the scope of neural networks is devoted to self organising maps, aimed at mapping incoming data vectors with different lengths onto one- or two-dimensional discrete map.

This work investigates the possibility of processing discrete time series of different duration by self-organising maps. One of the practical examples of such task is forecasting the product life cycle phase transition moment. The paper presents a model of the system for solving the aforementioned task.

2. Model of the system

The modular multiSOM system discussed below is intended for processing the discrete time series of different duration l that carry information about temporal changes of a target parameter value. As an example of such time series, the product demand information can be mentioned. Such time series will carry temporal changes of a product demand, providing information about the current and previous phases of product life cycle. Having this information, and also the data on transition moments between phases of the product life cycle, it becomes possible to generate key shapes (images) for transition moments that can be used to forecast the product life cycle phase transition moments for new products.

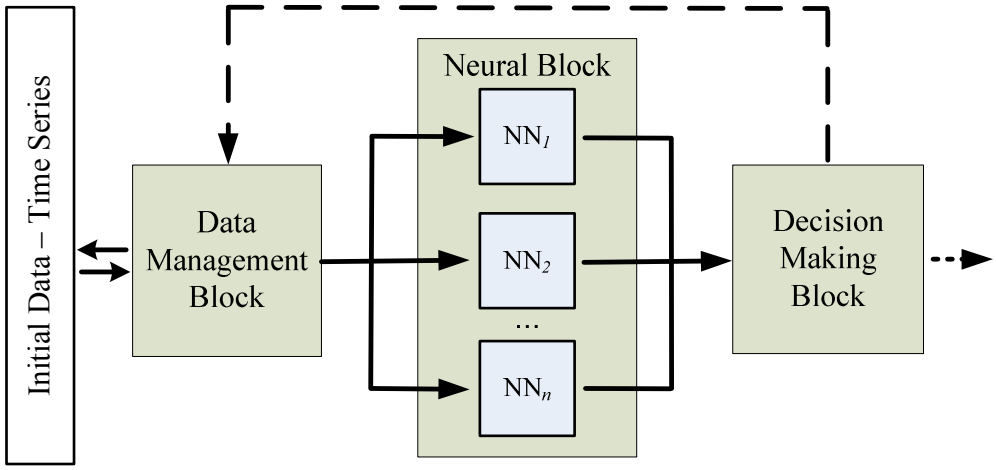

Figure 1. Structure of the system

Figure 1 displays the model of the system designed. The system contains three blocks: Data Management Block, Neural Block and Decision Making Block.

2.1. Data Management Block

The Data Management Block performs several tasks such as processing input data and spreading it between modules – self-organising maps, in the neural block according to the load distribution policy chosen.

The input dataset for the system contains discrete time series with different duration l appearing as a part of a discrete set L. Minimal - $l_{\min}$, and maximal - $l_{\max}$, values of l depend on the selected input dataset. The set L represents the total system load that should be separated among the modules of the system. The value of chosen individual load for neural network - q, affects a factor showing whether the neural network will process time series with equal duration l (while $q = 1$), or (while $2 \le q \le |L|,\ |L| \ge 2$) the single neural network will process time series with different durations.

The Data Management Block performs also the simulation of online data flowing into the system task. Such process is necessary due to imitation of real environment, where the system presented may be applied.

Data processing without imitation of data flowing, *Offline data processing* is based on this principle: at the moment when record R is ready to be sent to the system, the entire record is being sent to the corresponding module.

2.2. Neural Block

The Neural Block consists of a finite set of self-organising neural maps $M = \{m_1, m_2, ..., m_n\}$ where each neural net m_i is a module and according to chosen policy of system load spreading processes specific fraction of input data. The chosen policy of system load spreading affects the number of modules in the Neural Block, as well as the number of synaptic weights neurons will have in each neural network. Each module in the Neural Block is based on the Kohonen self-organising map [1, 2, 4].

The number of synaptic weights of a neuron in classical Kohonen map equals to the length of records in the input dataset. Due to such limitations, in the developed system it is possible to use the classical Kohonen map only when $q = 1$ for each neural network $m \in M$. For to be able to maintain the system functionality while $q > 1$, it is necessary to apply some modifications to classical Kohonen map.

In this work a heuristic, based on substituting the distance measure, is considered. We

propose substitution of a Euclidean distance measure with a measure based on Dynamic Time Warping. Using DTW allows one to process time series with different duration. Classical Dynamic Time Warping algorithm is described in [5, 6].

2.3. Decision Making Block

The Decision Making Block receives information from modules of Neural Block, activated by input data. Let us specify the set of clusters in the neural network $m_i \in M$ as C_{m_i}, $C_{m_i} = \{c_1, c_2, ..., c_j, ..., c_f\}$, and the set of neurons in the neural network $m_i \in M$ as N_{m_i}, $N_{m_i} = \{n_1, n_2, ..., n_k\}$. Neurons can be marked as clusters, but also may remain neutral in case if a neuron was never a winning neuron. Due to it, the assumption $|C_{m_i}| \le |N_{m_i}|$ will be correct.

Each signal sent from a neural module m_i to Decision Making Block brings information on the closest cluster c_j in module m_i for record R. This information gives a basis for a decision about a link between record R and a target value p.

3. Main algorithm

This section contains a description of the main algorithm, executed in three steps - System organisation; System testing and validation and Application of the system to the real world problem.

3.1. System training

The initial phase of system training process is determination and setting of basic system's parameters: the number of neuron networks (modules) in the system, dimensionality and topology of networks and the number of neuron connections in the networks.

The number of modules in a neural block, n, is calculated heuristically. Given a policy assuming uniform distribution of general load among the networks, formula (1) can be used to calculate the number of modules in the system:

$$n = \left\lceil \frac{|L|}{q} \right\rceil, \tag{1}$$

where q - each network's individual load; $\lceil \ \rceil$ - symbol of rounding up.

After the number of modules, n, is determined for each module m_i, an interval of the fixed length $\lfloor l_{\min_i}, l_{\max_i} \rfloor$ is set. The records of the fixed length, $l \in \lfloor l_{\min_i}, l_{\max_i} \rfloor$ will be processed by module m_i. Given a uniform load distribution, formula (2) can be used.

$$\begin{cases} i = 1, \quad l_{\min_i} = l_{\min} \\ i > 1, \quad l_{\min_i} = l_{\max_{i-1}} + 1, \\ l_{\max_i} = l_{\min_i} + q - 1 \end{cases} \tag{2}$$

The number of neurons of each neural network is determined heuristically depending on the task stated. The number of neuron connections in each network can be calculated using formula (3) below.

$$b_j = \left\lceil \frac{l_{\min_i} + l_{\max_i}}{2} \right\rceil, \tag{3}$$

3.1.1 Formation of self-learning neural networks

The algorithm for self-learning neural networks formation begins with initialisation of synaptic weights of each neuron. Normally it is accomplished by assigning to the synaptic

weights small values produced by random number generator. When this kind of neural network formation is employed, initially the network has no organization at all. Then the following main processes are launched: Competition, Cooperation and Synaptic adaptation [1, 4].

According to the selected process of data input, Data Management Block forwards each record R of the training set to the corresponding network m. Then the process of neuron competition for the right to become the winning or best-matching neuron for the arrived record begins. Discriminant function – the distance between the vector of the synaptic weights and discrete time series – is calculated using the DTW algorithm. Thus the neuron with the least total distance becomes the winner or best-matching neuron.

The winning neuron is located in the centre of the topological neighbourhood of co-operating neurons. Let us define lateral distance between the winning neuron (i) and re-excited neuron (j), as $d_{j,i}$. Topological neighbourhood $h_{j,i}$ is symmetric with regard to the point of maximum defined at $d_{j,i} = 0$. The amplitude of the topological neighbourhood $h_{j,i}$ decreases monotonically with the increase of lateral distance $d_{j,i}$, which is the necessary condition of neural network convergence [4]. To ensure the process of self-organisation, the synaptic neuron weights has to change in accordance with the input data, i.e. adapt to the input space. Let us assume that w_j(n) is the vector of synaptic weights of neuron j. In this case, at time instant $n+1$ the renewed vector $w_j(n+1)$ is calculated by formula (4).

$$w_j(n+1) = w_j(n) + \eta(n) \cdot h_{j,i(R)}(n) \cdot \left(A - w_j(n)\right), \tag{4}$$

where η - learning rate parameter; A - discrete time series in record R.

Note how the difference between discrete time series and the vector of synaptic weights is calculated in expression (4). When the load is $q = 1$, that is when each neural network is processing discrete time series with a certain fixed duration, and DTW is not used, the difference between A and $w_j(n)$ is calculated as the difference between vectors of different length. In other cases when DTW is employed, the fact of time warping has to be taken into account. For this purpose, it is necessary to fix in the memory a warping path on whose basis the distance between the vector of synaptic weights of the winner neuron and discrete time series was calculated. Thus the following information is becoming available: according to which value of the discrete time series the corresponding synaptic weight of the neuron has to be adjusted.

3.1.2 Cluster formation

As soon as the process of self-organising neural network training is finished, the process of cluster formation begins. Each record R of the training set is passed to the system. The same principle of data input (Online/Offline) is used as in organizing neural networks. Algorithms for module m_i and winner neuron n_j^* determination fully coincide with those used in network organization.

In parallel, for each neuron n_j^* these statistics are kept: records with which value of the key parameter p have got to neuron n_j^* and how many times. Cluster c_i is a neuron n_j that at least once became the winner neuron during cluster formation.

Once the last record of the training set is processed, for each cluster $c \in C_{m_i}, C_{m_i} = \{c_1, c_2, ..., c_f\}$ a base value of the key parameter p^* is defined which will be produced by the system as an output for record R that has got to the given cluster during system testing.

The simplest way to determine the base value of the key parameter p^* for cluster c_i is to select value $p = p'$ that has the highest frequency according to the previously collected statistics for values p in cluster c_i.

Situations might occur when a cluster has several possible values p'. As of today, in the system developed by the authors, if the above situation occurs, the least by module value p' is assumed as a base value of the key parameter. As applied to the task of product life cycle phase transition period forecasting, it means that the value will be chosen that forecasts transition in earlier period as compared to others. Also variants are possible, when out of several p' the largest by module value p' is chosen or the value p' closest to the median of distribution of values p fixed in the cluster.

3.2. System testing

To test the system, the same data input principle (Online/Offline) is used as in organizing neural networks.

Two criteria are employed to evaluate the effectiveness of the system: Mean Absolute Error – MAE) to evaluate the accuracy of the system and Logical Error to evaluate whether decisions made by the system are logically correct.

The Mean Absolute Error (MAE) is calculated using formula (5)

$$MAE = \frac{\sum_{i=1}^{k} |p_i - r|}{k}, i = [1, 2, \ldots, k], \quad (5)$$

where k – the number of records used for testing; p_i – real value of the key parameter for record i; r – the value of the key parameter forecasted by the system.

Logical error provides information about the logical potential of the system. To calculate the logical error, it is necessary to define logically correct and logically incorrect decisions. As applied to the task of forecasting product life cycle phase transition period, logically correct and logically incorrect decisions could be described as follows:

1. Assume that record R, entering the system has a fixed duration of the discrete time series A equal to l_A but the value of the key parameter - the period of product life cycle phase transition period is, $p = p_A$, where $p_A > l_A$. This statement means that a real time of transition between the phases of the product life cycle has not come yet. Accordingly, logically correct decision is to forecast transition period r_A, where $r_A > l_A$. Logically incorrect decision in this case will be if $r_A \le l_A$.
2. Assume that record R, that has entered the system has a fixed duration of the discrete time series A equal to l_A and the value of the key parameter - transition period of product life cycle phase, $p = p_A$, where $p_A \le l_A$. This statement gives evidence that real transition moment has already come. Accordingly, logically correct decision could be forecasting transition period r_A, where $r_A \le l_A$. In its turn, logically incorrect decision will take place if $r_A > l_A$.

The statement that at $r_A = l_A$ transition has occurred can be considered correct as the availability of data about some period in record R shows that the period is logically completed and, consequently, the transition -if any was assumed in this period- is also completed.

4. Experiments and results

This section shows an example of a practical realisation of the algorithms described in

the previous section and provides an analysis of the obtained experimental results. The system is built using the *Microsoft Visual Basic 2008 Express* programming environment.

4.1. Data and description of experiments

The data used to organise and test the system contain a demand for different products during the introduction phase plus one period of the maturity phase of the selected products. For each product the "Introduction to Maturity" transition period is supplied. The supplied dataset contains 199 discrete time series with minimal transition period equal to three ($l_{\min} = 4$) and maximal – in the 23rd period ($l_{\max} = 24$), giving $|L| = 21$. Since there is no data available about the absolute maximum and minimum of a demand, the discrete time series data was normalized using the *Z-score* normalization approach.

The main purpose of the experiments is to collect and analyse data about the changes of the efficiency of the described multi-SOM system depending on the chosen neural network load.

Neural network load q varies in the interval from $q = 1$ to the maximal possible within the selected dataset value, which is $q = 21$. The uniform distribution of the load among modules of the system is chosen as load distribution policy. Parameters that were not changed during the experiments are shown in Table (1)

Table 1

Static parameters of the neural networks

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

The process of system training is limited by 100 cycles. Within this number of cycles the system will not reach the state of complete or close to complete convergence. Nevertheless, training the system within 100 cycles is sufficient for discovering tendencies of system's effectiveness changes depending on the selected load q.

For each q the system was run 10 times, each time being tested with the help of the10 fold cross-validation method. Using the data collected, an average effectiveness of the system at a certain load q was calculated.

4.2. Results

This subsection presents the results achieved and an analysis of the experimental data collected. The experimental results are graphically displayed in Figures 2 and 3.

From the figures it can be seen that the variation of the mean absolute error with regard to the network load is close to linear but is not such. In its turn, in the tendency of logical error variation there can be observed a non-linearity. The highest accuracy of the system was reached at $q = 1$, i.e. when each neural network was processing discrete time series of one fixed duration. In the example described above, at $q = 1$ the number of neural networks (modules) in the system is equal to the number of elements in the discrete set L, i.e. $|M| = |L| = 21$. With the increase of the number of elements L the number of modules will grow in parallel and also the computational load. Such a tendency will inevitably lead to the critical moment, when the system will grow and will turn economically unprofitable for the user.

While analysing the obtained results in more detail, one can see that at certain values of q, say at $q = 4$, the accuracy of the system is close to that at $q = 1$ (see Figure 2). This makes it possible to conclude that neural network load q can be increased by a certain value without

any essential losses in the system's accuracy. The use of a single module for processing discrete time series of different duration enables one to enlarge the number of elements in discrete set L, upon reaching which the system will become economically unprofitable.

It should be noted that the determined values of q, at which the accuracy of the system is close to that at $q = 1$, depend on the training dataset used.

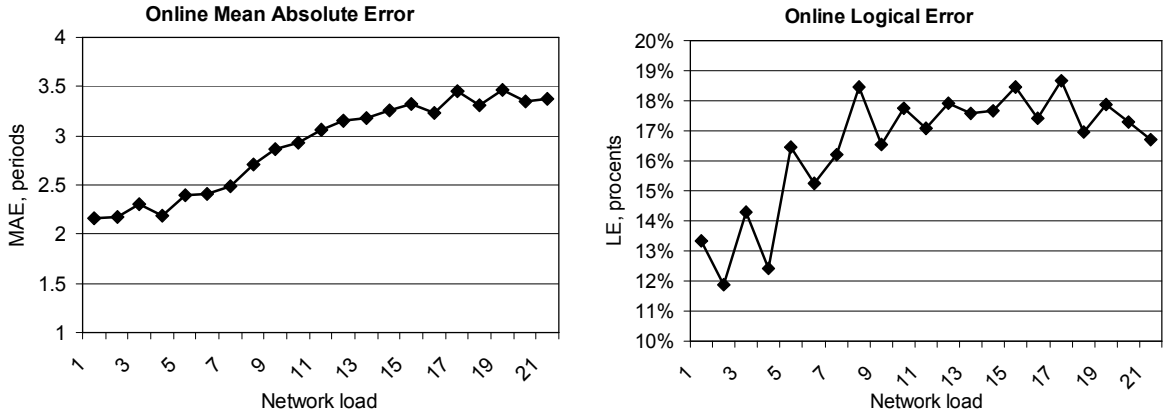

Figure 2. Online results

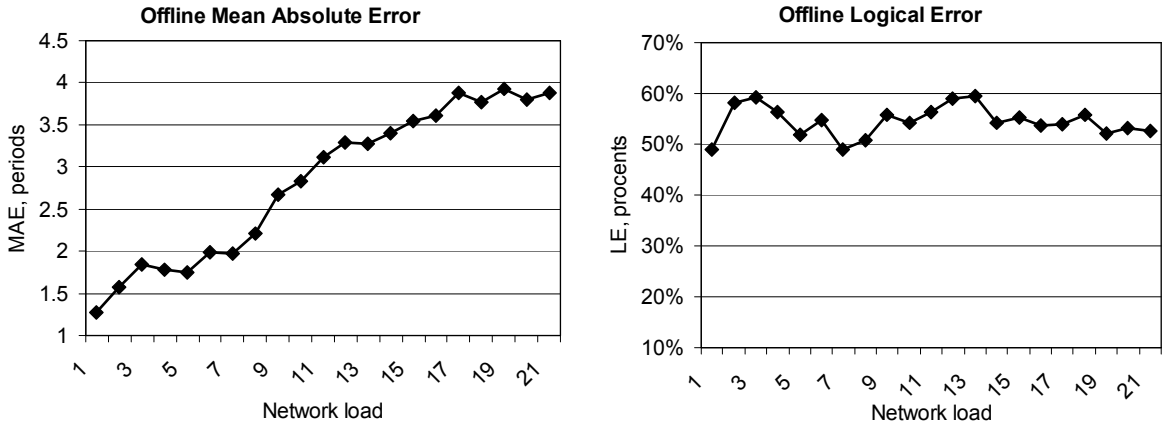

Figure 3. Offline results

5. Conclusions

In this paper a task of clustering time series of different duration is stated and a structure of data mining system based on modular neural networks is shown. As a practical example of the aforementioned task, the problem of forecasting product life cycle phase transition moment was chosen. The results obtained make it possible to conclude that using the DTW techniques in self-organising maps provides an option of processing time series of different duration with a single neural network. Besides, by choosing appropriate load (q) for the neural module it is possible to lessen the size of the system with minimal loss in efficiency.

The system uses the time series with target marker as an input and presents to the user the possible life cycle phase transition moment. The knowledge, describing in which phase the product currently is, proves topical for practitioners of management of product life cycle. Such knowledge helps in choosing between cyclic and non-cyclic supply chain managing policies.

Possible direction of future research could be the treatment of life cycle phase transition moment forecasting in the context of semi-supervised learning [8].

References

1. Kohonen T. Self-Organising Maps. Springer, 3rd edition, 2001, 501 p.
2. Self-Organising Map Formation. Edited by Obermayer K. and Sejnowski T. // The MIT Press, 2001, 440 p.
3. Tan P.-N., Steinbach M. and Kumar V. Introduction to Data Mining. Pearson Education, 2006, 769 p.
4. Haykin S. Neural Networks. // Prentice Hall, 2nd edition, 1999, 1104 p.
5. Chu S., Keogh E., Hart D. and Pazzani M. Iterative Deepening Dynamic Time Warping. // Second SIAM International Conference on Data Mining, 2002. Link: http://www.cs.ucr.edu/~eamonn/sdm-02.pdf, last viewed – September, 2008.
6. Keogh E. and Pazzani M. Derivative Dynamic Time Warping. // First SIAM International Conference on Data Mining, 2001. Link: http://www.cs.ucr.edu/~eamonn/sdm01.pdf, last viewed – September, 2008.
7. Parshutin S. and Kuleshova G. Time Warping Techniques in Clustering Time Series // Proceedings of 14th International Conference on Soft Computing, Mendel 2008, P.175-180.
8. Zhu X. Semi-supervised learning literature survey. // Technical report 1530, Department of Computer Sciences, University of Wisconsin, 2008. Link: http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf, last viewed – September 2008.

Paršutins Sergejs, Aleksejeva Ludmila un Borisovs Arkādijs. Laika rindu analīze ar moduļveida neironu tīklu palīdzību

Dotajā darbā ir demonstrēta moduļveida neirontīklu sistēmas struktūra, kas ļauj apstrādāt atšķirīga ilguma diskrētas laika rindas. Par praktisko uzdevumu sistēmas iespēju testēšanai tika izvēlēta produkta dzīves cikla fāzes mainīšanas momenta prognozēšanas problēma. Galvenais darba mērķis ir atšķirīga ilguma diskrēto laika rindu apstrādes iespējas ar vienu neironu tīklu – vienu sistēmas moduļi, pētīšana. Darbā tika veikti eksperimenti ar mērķi izpētīt sistēmas efektivitātes izmaiņas atkarībā no atšķirīga ilguma laika rindu skaita, kuru apstrādā atsevišķs neirontīkls. Minētais parametrs tika mainīts sākot ar gadījumu, kad katrs neirontīkls apstrādā laika rindas ar vienu noteikto ilgumu, un palielinājās līdz brīdim, kad sistēmā tika izveidots tikai viens neirontīkls, kas apstrādāja visas apmācības kopā pieejamas laika rindas. Pēc iegūtiem rezultātiem var secināt, ka atkarībā no sistēmas apmācībai izmantojamas datu kopas ir iespējams noteikt laika rindu ilguma intervālu katram sistēmas neirontīklu modulim, kas ļaus sistēmai funkcionēt ar pieņemamo efektivitāti, salīdzinot ar gadījumu, kad katrs sistēmas modulis ir spējīgs apstrādāt tikai viena ilguma diskrētas laika rindas.

Parshutin Serge, Aleksejeva Ludmila and Borisov Arkady. Time series analysis with modular neural networks

This work demonstrates the structure of the modular neural network based system for processing discrete time series of different duration. As a practical task for analysing and testing the system, the task of forecasting the product life cycle phase transition moment was chosen. The goal of this paper is to investigate the possibility of processing discrete time series of various durations by a single neural network – by a single neural module in the system. The work presents the experiments aimed at an investigation of system efficiency changes with regard to the number of time series of different durations each neural module can handle. The experimental results gained make it possible to conclude that depending on the chosen learning set it is possible to specify the interval of time series duration values for each neural module in the system, allowing the system to function on an appropriate level of efficiency comparing to the case when each module processes single time series duration value.

Паршутин Сергей, Алексеева Людмила и Борисов Аркадий. Анализ временных рядов с применением модульных нейронных сетей

В рассматриваемой работе представлена структура системы на основе модульных нейронных сетей, позволяющей обрабатывать временные ряды различной длительности. В качестве практической задачи для апробирования системы использована задача прогнозирования момента смены фазы жизненного цикла продукта. Основной целью работы является изучение возможности обработки дискретных временных рядов различной длительности с помощью одной нейронной сети – одного модуля в системе.

В рамках работы проведены эксперименты по изучению изменения эффективности системы в зависимости от количества временных рядов различной длительности, обрабатываемых одной нейронной сетью. По результатам экспериментов становится возможным заключить, что в зависимости от использованной для обучения системы выборки возможно выделение интервала длительностей дискретного временного ряда для каждого модуля, позволяющего системе сохранять эффективность на приемлемом уровне по сравнению с вариантом, когда каждый модуль способен обрабатывать временные ряды одной длительности.