Forecasting Product Life Cycle Phase Transition Points with Modular Neural Networks Based System

Serge Parshutin, Ludmila Aleksejeva, and Arkady Borisov

Riga Technical University, Institute of Information Technology, 1 Kalku Str., Riga, Latvia, LV-1658
serge.parshutin@rtu.lv, ludmila.aleksejeva@cs.rtu.lv, arkadijs.borisovs@cs.rtu.lv

Abstract. Management of the product life cycle and of the corresponding supply network largely depends on information in which specific phase of the life cycle one or another product currently is and when the phase will be changed. Finding a phase of the product life cycle can be interpreted as forecasting transition points between phases of life cycle of these products. This paper provides a formulation of the above mentioned task of forecasting the transition points and presents the structured data mining system for solving that task. The developed system is based on the analysis of historical demand for products and on information about transitions between phases in life cycles of those products.

The experimental results with real data display information about the potential of the created system.

Keywords: Modular Neural Networks, Self-Organizing Maps, Product Life Cycle, Forecasting Transition Points.

1 Introduction

Constantly evolving computer technologies are becoming more and more inherent part of successful enterprises management and keeping its activity at a high level. Different institutions are trying to reduce their costs by fully automatising certain stages of manufacturing process as well as introducing various techniques intended for forecasting certain market indicators that impact general manufacturing process. Different statistical methods are employed as well, though an increasing interest in artificial intelligence technologies and their practical application can be observed ever more.

For quite a long time neural networks have been one of the most popular research areas in the field of various processes forecasting including non-linear ones. The number of publications, books and monographs published within the last few years gives apparent evidence of that. A special place among neural networks is occupied by self-organising maps whose primary goal is to transform the incoming vectors of signals that are of deliberate dimensionality into single- or two-dimensional discrete map.

This paper focuses on studying self-organising map's ability to process discrete time series of different duration. A task of product life cycle phase transition point forecasting can serve as an example of analysis of different duration time-series. From the viewpoint of the management it is important to know, in which particular phase the product is. One of applications of that knowledge is selection of the production planning policy for the particular phase [10]. For example, for the maturity phase in case of determined demand changing boundaries it is possible to apply cyclic planning [2], whereas for the introduction and decline phase an individual planning is usually employed. This paper proposes a model of modular multi-network system that ensures the solving of the aforementioned task as well as provides an analysis of system testing results.

The paper is organised as follows: Section 2 formulates the task of forecasting a product life cycle phase transition point, followed by the Section 3 with structure of the created model presented and with functional aspects of the system elements described. The gathered experimental results are presented and analysed in Section 4 followed by conclusions.

2 Problem Statement

Any created product has a certain life cycle. The term "life cycle" is used to describe a period of product life from its introduction on the market to its withdrawal from the market. Life cycle can be described by different phases: traditional division assumes such phases like introduction, growth, maturity and decline [9]. For products with conditionally long life cycle, it is possible to make some simplification, merging introduction and growth phases into one phase - introduction.

An assumption that three different phases, namely, introduction, maturity and end-of-life are possible in the product life cycle, gives us two possible transitions. The first transition is between introduction and maturity phases and the second - between maturity and product's end-of-life.

From the side of data mining [3,4,5] information about the demand for a particular product is a discrete time series, in which demand value is, as a rule, represented by the month. A task of forecasting a transition points between life cycle phases may be formulated as follows. Assume that $D = \{d_1, \ldots, d_i, \ldots, d_n\}$ is a dataset and $d = \{a_1, \ldots, a_j, \ldots, a_l\}$ is a discrete time series whose duration equals to l periods, where $l \in L = \{l_1, \ldots, l_h, \ldots, l_s\}$ and varies from record to record in the dataset D. For simplification, the index of d is omited. Time series d represents a particular phase of a product life cycle, say introduction. Assume that for a particular transition, like introduction to maturity, a set of possible transition points $P = \{p_1, \ldots, p_k, \ldots, p_m\}$ is available. Having such assumptions the forecasting of a transition point for a new product, represented by a time series $d' \notin D$, will start with finding an implication between historical data sets D and P, $f : D \to P$; followed by application of found model to new data.

3 Model of the Modular Neural Networks Based System

This section contains a description of the general structure of the proposed system as well as provides information about the functions and importance of each block of the system.

3.1 Structure of the System

The main distinctive feature of the developed modular multi-network system (multiSOM system) is its ability to process discrete time series of different duration l, that represent information about the changes in certain indicator over time, e.g. previously mentioned demand for a product.

In general, the format of historical input data that could be processed by the system has to comply with these conditions:

– Each record displays the demand for a product, collected within known period of time, the length of which is set by the system - day, week, month, etc. In other words, each record is a demand time series.
– Each record has one or both markers - transition indicators:
  • marker *M1* indicates the period when product switched from Introduction phase to Maturity phase;
  • marker *M2* indicates the period when product switched from Maturity phase to End-of-Life phase.
– Each record has a marker indicating the moment of the actual beginning of the Introduction phase (ABI).

The last condition is based on the fact that in many databases records are kept from the defined moment in time. It is evident that not all historical products were introduced on the market at the same moment in time. Marks on transitions can guarantee that a model will be build; if we have patterns of transitions in historical data, then, theoretically, in presence of a model for generalisation, we are able to recognise those patterns in new data. To lessen the impact of noisiness and dominance of data, the dataset should be preprocessed. Data should be normalized and obvious outliers should be excluded [13,14].

Figure 1 shows a structure of the proposed system consisting of three main blocks: Data Management Block, Neural Block and Decision Making Block.

Data Management Block. The Data Management Block (DMB) performs tasks of processing input data and their distribution over modules in Neural Block. Data distribution over neural modules occurs in accordance with load distribution policy specified by the user. Let us define total system load as the number of elements in set L. At the given distribution, load distribution policy shows which part of general load will be taken by each neural module. The chosen distribution directly affects the number of neural modules in the system.

Let us illustrate load distribution. Assume that the duration of discrete time series in dataset D varies from 4 to 11 periods, thus $l \in L = \{4, 5, \ldots, 11\}$. In this

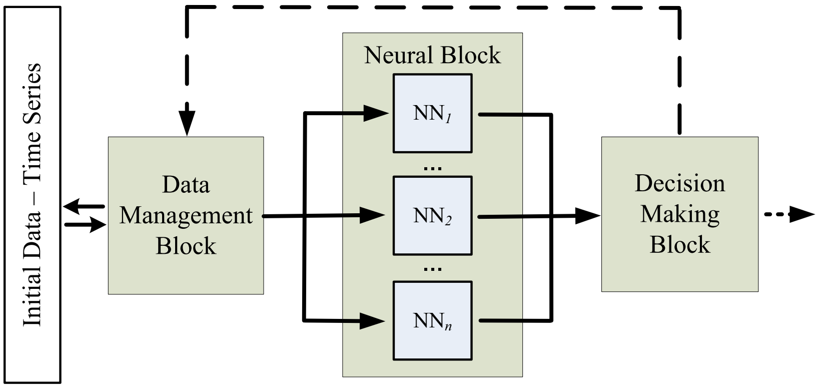

Fig. 1. Structure of the system

case, total system load will consist of eight values that time series duration can take. Let us assume that the load has to be distributed uniformly over modules at the condition that an individual load on separate module should not exceed three values that is $q = 3$. Under such conditions, three neural modules will be created in the Neural Block by the moment of system initialisation. The first neural module will process time series of duration $l \in \{4, 5, 6\}$; records with $l \in \{7, 8, 9\}$ will be sent to the second module. The remaining time series with duration of 10 and 11 periods will be processed by the third neural module.

In the real world situation, the information about the demand for a new product is becoming available gradually: after a regular period finishes, new demand data appear. Due to that specifics of system application environment, at the stage of learning it is necessary to use On-line data flowing imitation procedure. This procedure is implemented at the DMB level. The algorithm employed is executed taking into account the following details. Time series d contains demand data within introduction or maturity phase; it has duration l and is connected with the appropriate marker (*M1* or *M2* depending on a represented phase of a product life cycle) with value p. Provided that the system is able to process discrete time series with durations equal to l_{min} and greater, the algorithm of On-line data flowing procedure will include these steps:

1. Define $l^* = l_{min}$;
2. Send to the system first l^* periods of record d and a marker value p;
3. If $l^* < l$ then increase value of l^* by one period and return to step 2; else proceed to step 4;
4. End processing of record d.

Figure 2 illustrates execution of On-line data flowing procedure at $q = 3$.

Data processing without imitation of data flowing, *Off-line data processing* is based on this principle: at the moment when record d is ready to be sent to the system, the entire record with marker p is being sent to the corresponding module.

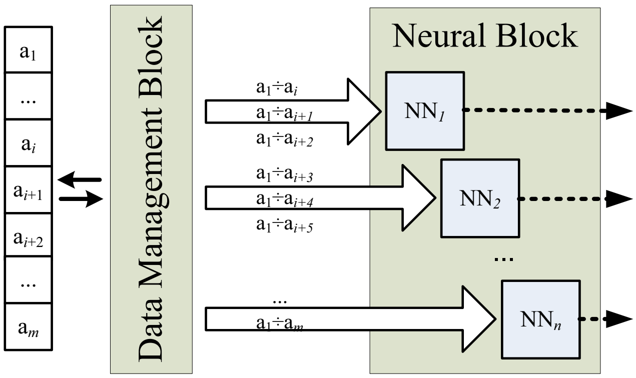

Fig. 2. On-line data flowing

Neural Block. According to the system structure depicted in Figure 1, DMB directs the data further to the corresponding modules of Neural Block (NB). Each module is a self-organizing neural map; according to the chosen policy of system load distribution it processes specific fraction of input data. The number of modules in the Neural Block, as well as the number of synaptic weights neurons will have in each neural network, is also affected by the chosen policy of system load distribution.

Each self-organising map is based on the modified Kohonen map. The necessity to modify the classical Kohonen map appears because of the stated task to use the same SOM for processing discrete time series of different duration. The number of synaptic weights of a neuron in the classical Kohonen map equals to the length of records - the duration of time series, in the input dataset [6,8,11,1]. Due to such limitations, it is possible to use the classical Kohonen map in the developed system only when $q = 1$ for each neural network. For to be able to maintain the system functionality while $q > 1$, it is necessary to apply some modifications to the 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, that is applied in the system, is fully described in [7]. The Dynamic Time Warping algorithm was experimentally compared with Derivative Dynamic Time Warping algorithm in the scope of the task, defined in Section 2, (see source [12]). The gathered results speak in favour of using the DTW for the specified task.

Functional aspects of NB are as follows: in the NB neural maps are organised at the stage of learning after that clusters are formed in each map; whereas at the stage of system application NB determines object's -record's or time series'- membership in one of previously formed clusters and forwards information from cluster further according to the scheme of transition point determination.

Decision Making Block. Decision Making Block is the one that receives information from the Neural Block on the best matching cluster found for a particular object. Following certain algorithms, whose examples are provided in the next section, a decision is shaped regarding a current object. The decision can contain both a particular value and a percentage distribution among several possible values of transition points. Taken into account that the final decision is made by a decision making person, the second option proves to be most preferable.

3.2 Basic Algorithm of System Functioning

The basic system functioning algorithm consists of the following three main stages: System training; System testing and validation and Application of the system to the real world problem. These stages are graphically displayed in Figure 3 and as can be seen from the scheme, the system functioning algorithm is cyclic.

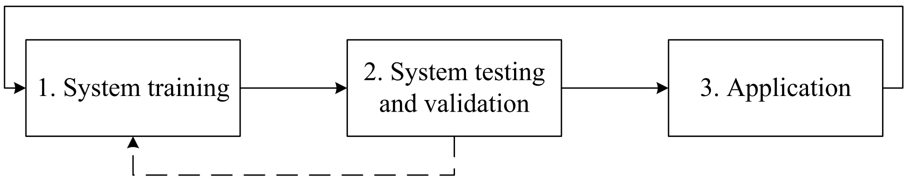

Fig. 3. General steps of the algorithm

As the system is employed, new data are accumulated an increase in whose amount will inevitably lead to the necessity to update the system. This process is initiated by feedback from step 3 to step 1. In its turn, feedback from step 2 to step 1 enables entering necessary corrections into the system even at the testing stage, which is more convenient from the viewpoint of management as compared to error correction after the application of the system to the real world problem. The following current subsection considers each of the three steps in more detail.

Step 1. System Training. The initial phase of system training process is determination and setting of basic system's parameters: the number of neural networks (modules) in the system, dimensionality and topology of networks and the number of synapses each neuron will have in each network. The number of modules in a neural block, n, is calculated empirically. 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; $\lceil \cdot \rceil$ - symbol of rounding up.

In parallel with calculating the number of modules in the system, it is necessary to determine a topology of neuron dislocation in the network. Several frequently used topologies can be applied [8,1] - tetragonal, hexagonal and a combination of the first two mentioned (see Figure 4).

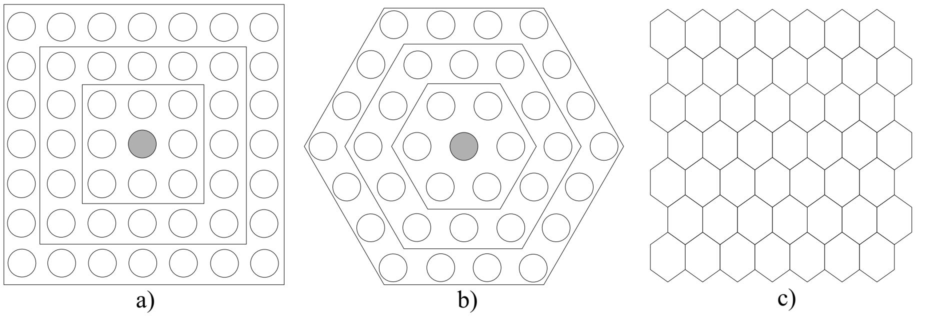

Fig. 4. Network topologies: a) - tetragonal; b) - hexagonal; c) - a) and b) combination

After the number of modules, n, is determined, for each module m_i an interval of time series durations $[l_{i,min}; l_{i,max}]$ is set. The records with duration $l \in [l_{i,min}; l_{i,max}]$ will be processed by module m_i. Given a uniform load distribution, equation (2) can be used:

$$\begin{cases} i = 1, & l_{i,min} = l_{min} \ , \\ i > 1, & l_{i,min} = l_{i-1,max} + 1 \ ; \end{cases} \\ l_{i,max} = l_{i,min} + q - 1 \ . \tag{2}$$

The number of neurons of each network is determined empirically depending on the task stated. The number of synapses of a neuron in each network can be calculated using formula (3) below:

$$b_{j,i} = \left\lceil \frac{l_{i,min} + l_{i,max}}{2} \right\rceil \ , \tag{3}$$

where $b_{i,j}$ is the number of synapses of a neuron j in the network m_i.

As an alternative, a median calculation can be used for determining the number of synapses of a neuron. Assuming that module m_i can process discrete time series of duration $l \in [l_1, l_2, \ldots, l_i, \ldots, l_k]$, let us denote for each value l the number of records in the training set having duration equal to l, as $f \in [f_1, f_2, \ldots, f_i, \ldots, f_k]$. By having such an assumption, a median of time series durations a module m_i can process, may be calculated with formula (4):

$$Median = \frac{\sum_{i=1}^{k}(l_i \cdot f_i)}{\sum_{i=1}^{k} f_i} \ . \tag{4}$$

The calculated median must be rounded to the integer thus obtaining the number of synaptic connections of a neuron. Will the median be rounded to a smaller or a greater value, to a large extent depends on the task to be solved.

As the main parameters are set, the initialisation of the system begins. Synaptic weights of each neuron in each network are assigned initial values - usually small values produced by random number generator. At this moment networks in the system have no organization at all. Then the following main processes are launched: Competition, Cooperation and Synaptic adaptation.

According to the selected process of data input, Data Management Block forwards each record of the training set to the corresponding network. Then the process of neuron competition for the right to become the winner 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 winner neuron is located in the centre of the topological neighbourhood of co-operating neurons. Let us define lateral distance between the winner neuron (i) and and re-excited neuron (j), as $ld_{j,i}$. Topological neighbourhood $h_{j,i}$ is symmetric with regard to the point of maximum defined at $ld_{j,i} = 0$. The amplitude of the topological neighbourhood $h_{j,i}$ decreases monotonically with the increase of lateral distance $ld_{j,i}$, which is the necessary condition of neural network convergence [6]. Usually a Gaussian function if used for $h_{j,i}$ calculation (formula 5).

$$h_{j,i(d)} = \exp\left(-\frac{ld_{j,i}^2}{2 \cdot \sigma^2(n)}\right) \quad . \tag{5}$$

A decrease in the topological neighbourhood is gained at the expense of subsequent lessening the width of σ function of the topological neighbourhood $h_{j,i}$. One of possible kinds of value σ dependence on discrete time n is an exponential decline (formula 6).

$$\sigma(n) = \sigma_0 \cdot \exp\left(-\frac{n}{\tau_1}\right) \quad n = 0, 1, 2, \ldots \quad , \tag{6}$$

where σ_0 is the beginning value of σ; τ_1 - some time constant, such as the number of learning cycles.

To ensure the process of self-organisation, the synaptic weights of a neuron 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 at time moment (iteration, cycle) n. In this case, at time instant $n+1$ the renewed vector $w_j(n+1)$ is calculated by formula (7):

$$w_j(n+1) = w_j(n) + \eta(n) \cdot h_{j,i(d)}(n) \cdot (d - w_j(n)) \quad , \tag{7}$$

where η - learning rate parameter; d - discrete time series from learning dataset.

Note how the difference between discrete time series and the vector of synaptic weights is calculated in expression (7). 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 d and $w_j(n)$ is calculated as the difference between vectors of equal 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.

The network organization contains two main processes - initial organization followed by convergence process. The initial organisation takes about 1000 iterations. During this process each network gets an initial organization, the learning rate parameter decreases from 0.1, but remains above 0.01.

The number of iterations the convergence process takes is at least 500 times larger than the number of neurons in the network. The main difference from the initial organization is that during the convergence process the topological neighbourhood of a winner neuron contains only the closest neighbours or just the winner neuron.

As soon as the process of self-organising neural network training is finished, the process of cluster formation begins. Each record d of the training set is passed to the system. The same principle of data input (On-line/Off-line) 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 $c \in C, C = \{c_1, c_2, \ldots, c_f\}$ a base value of the key parameter p^* is defined which will be produced by the system as an output for record d 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.

Step 2. System Testing and Validation. To test the system, the same data input principle (On-line/Off-line) 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 (8):

$$MAE = \frac{\sum_{i=1}^{k} |p_i - r|}{k} \quad i = [1, 2, \ldots, k] \ , \tag{8}$$

where k - the number of records used for testing; p_i - real value of the key parameter for record d_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 discrete time series d, entering the system has a duration equal to l_d but the value of the key parameter - the period of product life cycle phase transition, is $p = p_d$, where $p_d > l_d$. 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_d, where $r_d > l_d$. Logically incorrect decision in this case will be if $r_d \leq l_d$.
2. Assume that discrete time series d, entering the system has a duration equal to l_d but the value of the key parameter - the period of product life cycle phase transition, is $p = p_d$, where $p_d \leq l_d$. This statement gives evidence that real transition moment has already come. Accordingly, logically correct decision could be forecasting transition period r_d, where $r_d \leq l_d$. In its turn, logically incorrect decision will take place if $r_d > l_d$.

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

Step 3. System Application. Application of the system means not only using a model, but also monitoring the performance of the system and collecting the new data. As the new data is collected, the steps of the main algorithm should be repeated (see Figure 3), and the system (neural modules) must be reorganized.

4 Experiments and Gathered Results

The fact that the data describes real life process and marks of transitions were putted by experts implies that some noisiness in data is present.

The obtained dataset contains 312 real product demand time series with minimal duration equal to 4 and maximal - to 24 periods. Each time series contains the demand during the introduction phase of a specific product plus one period of the maturity phase, and is marked with *M1* marker. Table 1 presents an example of data after the normalization process is finished. The first 11 periods are given. To normalize the data, the Z-score with standard deviation normalization method was applied. As the true bounds of the demand data in the dataset are unknown and the difference between values of various time series is high, the chosen normalization method is one of the most suitable ones.

Table 1. Example of normalized data (first 11 periods)

ID	*M1*	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
1	3	-4.620	0.060	-0.127	0.247							
2	8	-0.493	-0.549	-0.361	-0.607	-0.623	-0.596	1.592	1.637	-2.817		
3	23	-1.790	-1.664	-1.352	-1.070	-0.893	-0.807	-0.886	-0.547	-0.683	-0.149	-0.133
4	17	-0.969	-0.262	-0.800	0.044	-0.545	-0.169	-0.491	-0.329	-1.078	-0.188	-1.147
5	8	1.625	1.582	1.512	1.872	1.723	1.785	-0.623	-0.581	-0.403		
6	23	0.454	-0.092	-0.492	1.009	-2.080	-0.741	0.908	-2.454	-0.549	0.870	0.309

Figure 5 displays all periods for the same time series as presented in Table 1. As can be seen, the time series differs not only in duration, but also in amplitude and its pattern.

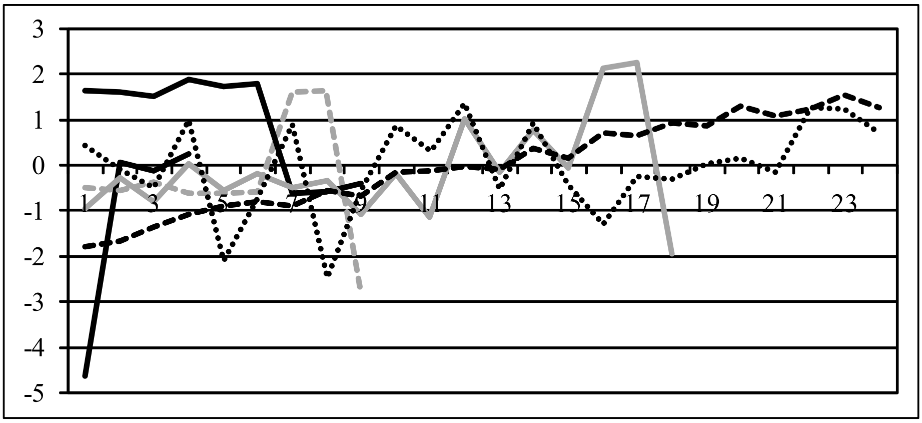

Fig. 5. Example of normalized data

The main target of the performed experiments was to analyse comparatively the precision of the system with different neural network topologies applied while using different network load q. The system was tested by sequentially applying each of three mentioned neural network topologies (see Figure 4) while network load q was changing incrementally from one to five. To calculate the system errors - Mean Absolute Error (MAE) and Logical Error (LE), a 10-fold cross validation method was applied, totally giving 150 system runs.

Table 2 contains the number of neurons in a neural network for each of three topologies, as also supplies the number of training and convergence cycles.

Table 2. Network and learning parameters

Topology	Neurons	Training cycles	Convergence cycles
a	25	1000	12500
b	37	1000	18500
c	25	1000	12500

The learning parameters, used for network organisation in each run, are given in Table 3. For each learning parameter the starting value and the minimal (last) value are supplied, as also the type of a function used for managing the parameter decline process.

Table 3. Learning parameters

Parameter	Starts with	Ends with	Function
Learning coefficient - η	0.9	0.01	Exponential
σ for Gaussian neighbourhood	0.5	0.01	Exponential

While testing the system in Online mode, for each neural network topology and for each of five defined values of q a Mean Absolute Error and a Logical Error were obtained. The gathered results are accumulated in Table 4 and Table 5, and graphically displayed in Figures 6 and 7 respectively.

Table 4. On-line Mean Absolute Error - MAE

Topology	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
a	2.917	2.950	2.940	3.186	3.042
b	3.061	2.912	3.151	2.989	3.152
c	3.066	3.148	3.166	3.267	3.102

The obtained results show that created system with certain precision is able to predict transition points for new products, using a model, built on a historical data. In On-line mode System was able to make a logically correct (refer to "System Testing and Validation" step in subsection 3.2) decision in at least 82.7% and at most in 86.5% of times. Thus the Mean Absolute Error lies close to three periods. Such an error may be unacceptable for a system, a forecast of

Table 5. On-line Logical Error - *LE*

Topology	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
a	16.1%	13.5%	13.9%	15.3%	14.2%
b	17.3%	14.7%	15.3%	16.2%	15.4%
c	16.6%	15.4%	15.6%	16.0%	14.0%

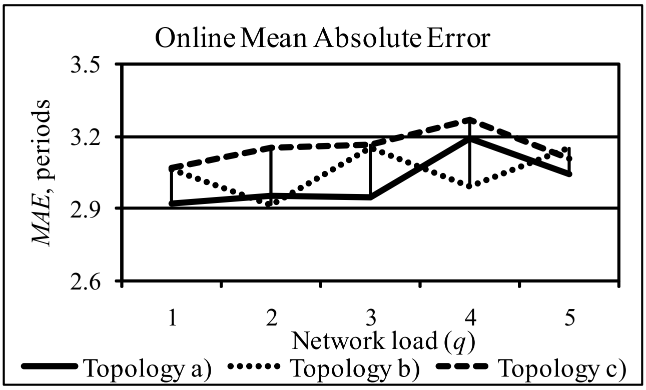

Fig. 6. On-line Mean Absolute Error

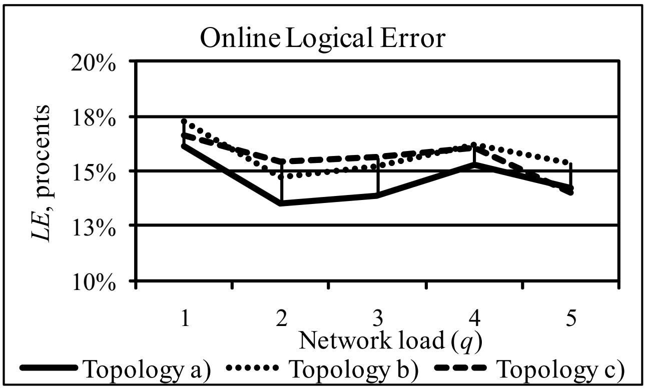

Fig. 7. On-line Logical Error

which will be just accepted without any further processing. Again mentioning the specificity of the dataset obtained and also bringing the point that the final decision is made by a decision making person, it is possible to conclude that the created system can be used as a data mining tool to gain an additional knowledge for solving a product life cycle phase transition points forecasting task, as well as other tasks, connected with forecasting a value of a target parameter for a time dependent variable.

5 Conclusions

For the practitioners of management of the product life cycle the knowledge, which describes in which phase the product currently is and when the transition between phases will occur, is topical. Such knowledge, in particular, helps to select between the cyclic and non-cyclic policy of planning supply chain operation.

In this paper, the task of forecasting the transition points between different phases of product life cycle is stated, and the structure of data mining system, which helps to solve this task, is shown. On the basis of the analysis of historical demand data for products it is possible to learn the modular neural network based system, which will be able to forecast the transition points in life cycle of new products. Experimentally gathered results show that the created system has its potential and can process real demand data, returning a forecast with a certain precision.

One aspect is that in the future it is necessary to examine the developed system on the data from different production fields, and, which is also important, to have a response from practitioners of supply chain management who will use these systems. Also relevant is to consider and analyse the possibility of using other Data Mining techniques in in the place of the Self-Organising maps.

Another aspect, modest data volume that was used for practical experiments, is related to the fact, that it is necessary to have transition marks in historical data from experts and practitioners. The more products, the more complicated for human to make all these marks - in practice the amount of marked data will always be restricted. As a result, possible direction of future research is treatment of forecasting the transition points in the context of a semi-supervised learning [15]. In this case, there is a small set with marked transitions and also a large dataset in which transitions are not marked. In such a situation it is necessary to create a model, which will be able to apply the knowledge, gathered on the small set of marked data, to the new (test) data.

Acknowledgements

The present research was partly supported by the ECLIPS project of the European Commission "Extended collaborative integrated life cycle supply chain planning system", and also supported by ESF Grant 2004/0002/VPD1/ESF/PIAA/04/NP/3.2.3.1/0001/0002/0007.

References

1. Agarwal, P., Skupin, A. (eds.): Self-Organising Maps: Application in Geographic Information Science. John Wiley & Sons Ltd., Chichester (2008)
2. Campbell, G.M., Mabert, V.A.: Cyclical schedules for capacitated lot sizing with dynamic demands. Management Science 37(4), 409–427 (1991)

3. Dunham, M.: Data Mining Introductory and Advanced Topics. Prentice-Hall, Englewood Cliffs (2003)
4. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufman, San Francisco (2006)
5. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge (2001)
6. Haykin, S.: Neural Networks, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
7. Keogh, E., Pazzani, M.: Derivative dynamic time warping. In: Proceedings of the First SIAM International Conference on Data Mining, Chicago, USA (2001)
8. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2001)
9. Kotler, P., Armstrong, G.: Principles of Marketing, 11th edn. Prentice-Hall, Englewood Cliffs (2006)
10. Merkuryev, Y., Merkuryeva, G., Desmet, B., Jacquet-Lagreze, E.: Integrating analytical and simulation techniques in multi-echelon cyclic planning. In: Proceedings of the First Asia International Conference on Modelling and Simulation, pp. 460–464. IEEE Computer Society Press, Los Alamitos (2007)
11. Obermayer, K., Sejnowski, T. (eds.): Self-Organising Map Formation. MIT Press, Cambridge (2001)
12. Parshutin, S., Kuleshova, G.: Time warping techniques in clustering time series. In: Proceedings of 14th International Conference on Soft Computing MENDEL 2008, pp. 175–180. Brno University of Technology (2008)
13. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann Publishers, an imprint of Elsevier (1999)
14. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education, London (2006)
15. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin (2008)