Analysis and Modeling of Complex Systems Using the Self-Organizing Map

Olli Simula, Juha Vesanto, Esa Alhoniemi, and Jaakko Hollmén

Helsinki University of Technology
Laboratory of Computer and Information Science
P.O. Box 2200, FIN-02015 HUT, Finland

Abstract The Self-Organizing Map (SOM) is a powerful neural network for analysis and visualization of high-dimensional data. It maps nonlinear statistical relationships between high-dimensional input data into simple geometric relationships on a usually two-dimensional grid. The mapping roughly preserves the most important topological and metric relationships of the original data elements and, thus, inherently clusters the data. The need for efficient data visualization and clustering is often faced in various engineering problems. In this chapter, SOM based methods are applied in analysis, monitoring and modeling of complex systems.

1. Introduction

In modeling and control of complex systems, it is usually assumed that a global, analytical system model can be defined. However, many industrial processes are so complicated that a global model cannot be built. In such cases, Artificial Neural Networks (ANNs) can successfully be used. ANNs build models directly based on process measurements, and thus provide a means to analyze processes without explicit physical process model. ANNs can also be used as "soft sensors" to estimate signal values or process variables that can only be measured indirectly or off-line. The use of the ANNs, however, requires that a large amount of good quality, stable, numerical data describing the process are available.

The Self-Organizing Map (SOM) [12] is a neural network algorithm which is based on unsupervised learning. Unlike supervised learning methods, the SOM can be used for clustering data without knowing the class memberships of the input data. It can, thus, be used to detect features inherent to the problem. The SOM has been successfully applied in various engineering applications [14] covering, for instance, areas like pattern recognition, image analysis, process monitoring and control, and fault diagnosis [28, 29, 34]. The SOM has also proven to be a valuable tool in data mining and knowledge discovery with applications in full-text and financial data analysis [8].

In this chapter, SOM based methods in the analysis of complex systems are discussed. Special emphasis is on industrial applications in which a lot of measured information is available from automation systems.

2. The Self-Organizing Map

2.1 Network Structure and Algorithm

The SOM consists of a regular, usually two-dimensional, grid of neurons. Each neuron i of is represented by a weight, or model vector, $\boldsymbol{m}_i = [m_{i1}, \ldots, m_{in}]^{\mathrm{T}}$ where n is equal to the dimension of the input vectors. The set of weight vectors is called a codebook.

The neurons of the map are connected to adjacent neurons by a neighborhood relation, which dictates the topology of the map. Usually rectangular or hexagonal topology is used. Immediate neighbors belong to the neighborhood N_i of the neuron

i. In the basic SOM algorithm, the topological relations and the number of neurons are fixed from the beginning. The number of neurons may vary from a few dozens up to several thousands. It determines the granularity of the mapping, which affects the accuracy and generalization capability of the SOM.

During iterative training, the SOM forms an elastic net that folds onto the "cloud" formed by input data. The net tends to approximate the probability density of the data: the codebook vectors tend to drift there where the data are dense, while there are only a few codebook vectors where the data are sparse [12].

At each training step, one sample vector $\boldsymbol{x}$ is randomly chosen from the input data set. Distances (i.e., similarities) between the $\boldsymbol{x}$ and all the codebook vectors are computed. The best-matching unit (BMU), denoted here by c, is the map unit whose weight vector is closest to the $\boldsymbol{x}$:

$$||\boldsymbol{x} - \boldsymbol{m}_c|| = \min_i \{||\boldsymbol{x} - \boldsymbol{m}_i||\}. \tag{2.1}$$

After finding the BMU, the weight vectors are updated. The BMU and its topological neighbors are moved closer to the input vector in the input space. The update rule for the weight vector of unit i is:

$$\boldsymbol{m}_i(t+1) = \begin{cases} \boldsymbol{m}_i(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)], & i \in N_c(t) \\ \boldsymbol{m}_i(t), & i \notin N_c(t) \end{cases} \tag{2.2}$$

where t denotes time. $N_c(t)$ is a non-increasing neighborhood function around the winner unit c and $0 < \alpha(t) < 1$ is a learning coefficient, which is a decreasing function of time.

The SOM algorithm performs a topology preserving mapping from the high-dimensional input space onto map units so that relative distances between data points are preserved. Data points lying near each other in the input space will be mapped onto nearby map units. The SOM can thus serve as a clustering tool of high-dimensional data. It also has capability to generalize, i.e. the network can interpolate between previously encountered inputs.

A schematic illustration of applying the SOM in industrial process analysis is shown in Fig. 2.1. The different stages in the figure are: (1) data processing (acquisition, preprocessing, feature extraction, normalization), (2) map training, (3) validation and interpretation and (4) visualization.

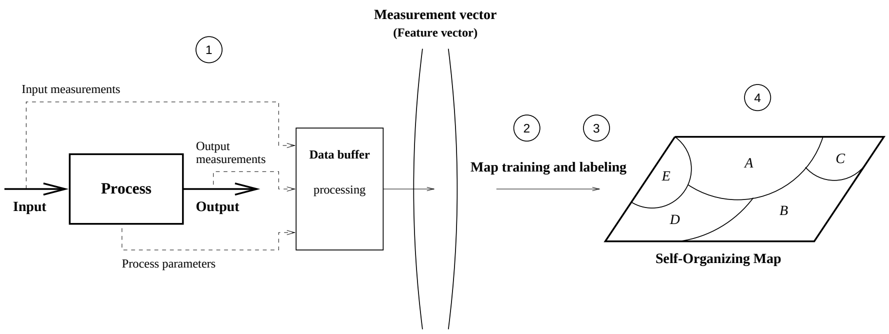


Figure 2.1. Stages of the application of the SOM in industrial process analysis.

2.2 Validation and Interpretation

The quality of the mapping is usually determined based on (1) precision and (2) topology preservation. The former can be measured using average quantization error, which is the average distance between the input vectors of the testing set and the corresponding BMUs. Different topology measures have been studied e.g. by Kiviluoto [11] and Kaski and Lagus [9]. The latter proposed a goodness meter measuring both precision and topology at the same time.

The SOM can be interpreted by naming the units according to input vectors, whose type (e.g., class) is known. This labeling gives physical interpretation of the network. If labeled vectors are not available, the map can be interpreted by direct inspection of the weight vectors and clusters on the map. This is easiest to accomplish using different visualization techniques discussed in detail in the next section. Also automatic interpretation of the map is possible using fuzzy rules as done by Pedrycz and Card [22].

2.3 Visualization

The SOM can be efficiently used in data visualization due to its ability to approximate the probability density of input data and to represent it in two dimensions. In the following, several ways to visualize the network are introduced using a simple application example, where a computer system in a network environment was measured in terms of utilization rates of the central processing unit and traffic volumes in the network. The SOM was used to form a representation of the characteristic states of the system.

The unified distance matrix (u-matrix) method by Ultsch [36] visualizes the structure of the SOM. Firstly, a matrix of distances (u-matrix) between the weight vectors of adjacent units of a two-dimensional map is formed. Secondly, some representation for the matrix is selected, for example, a grey-level image [7]. The u-matrix of the example system is shown in Fig. 2.2a. The lighter the color between two map units is, the smaller is the relative distance between them. On the left side, there is a large uniform area, which corresponds to idle state of the computer system. Top right corner forms a clearly separated area, which corresponds to high CPU load in the system.

Component plane representation visualizes relative component values of the weight vectors. The illustration can be considered as a "sliced" version of the map, where each plane shows the distribution of one weight vector component. Using the distributions, dependencies between different process parameters can be roughly studied. For example, Tryba *et al.* [35] have used this kind of visualization to investigate parameter variations in VLSI circuit design.

The component planes of the example system are presented in Fig. 2.2c. The colors of map units have been selected so that the lighter the color is, the smaller is the relative component value of the corresponding weight vector. It can be seen, for instance, that the components #1, #2 and #6 (read blocks per second, written blocks per second and write I/O percentage) are highly correlated.

Sammon's mapping is an iterative algorithm to project high-dimensional vectors in two dimensions [20]. The nonlinear mapping tries to preserve the relative distances between input vectors. The algorithm can be used to visualize the SOM by mapping the values of the weight vectors onto a plane. To enhance the net-like look, the neighboring map units may be connected to each other with lines to show the topological relations. Since the SOM tends to approximate the probability density of the input data, the Sammon's mapping of the SOM can be used as a very rough approximation of the form of the input data. The Sammon's mapping of the example system is illustrated in Fig. 2.2d. According to the mapping, the SOM seems to be well-ordered in the input space.

Naturally Sammon's mapping can be applied directly to data sets, but it is computationally very intensive making the algorithm too slow for large data sets. However, the SOM quantizes the input data to a small number of weight vectors, which lightens the burden of computation to an acceptable level.

Data histogram shows how input data are clustered by the SOM. In other words, it shows how many input vectors belong to clusters defined by each map unit. The histogram is formed using a trained SOM and a data set: for each data set vector, the BMU is determined, and the "hit counter" of that unit is increased by one. The histograms may be visualized in many ways. In our example, we have used squares of different sizes: the larger the counter value is, the larger is the square. The data histogram of the example application is shown in Fig. 2.2e.

Operating point and trajectory can be used to study the behavior of a process in time. The operating point of the process is the BMU of the current measurement vector. The location of the point on the topologically ordered map can be easily visualized and used to determine the current process state. If also the history of the process is of interest, a sequence of operating points in time forming a trajectory can be studied. The trajectory shows the movement of the operating point, which in some cases may be a very useful piece of information. A trajectory of the example system is illustrated in Fig. 2.2b. The trajectory starts from the normal operation area and moves through disk intensive phase to high load area.

2.4 Software Tools

`SOM_PAK`. The SOM algorithm has been implemented in a software package `SOM_PAK` [13] written in ANSI C. The package includes tools for SOM training, validation and visualization. The software uses simple ASCII files and can use data with missing vector components [27]. Graphical output of the program is in encapsulated postscript (`eps`) format. An exhaustive description of the package can be found in [13].

`SOM Toolbox`. Since the flexibility of the SOM can only be fully exploited in a versatile computing environment, a software package `SOM Toolbox` [2] has been developed for Matlab 5 environment [19]. The package includes basic initialization, training and validation algorithms of the SOM. Special attention has been given to easy and effective SOM visualization. The Figs. 2.2a – 2.2e in the Sect. 2.3 were mostly produced using the `SOM Toolbox`. Since all the algorithms have been implemented using simple and well documented Matlab scripts, the functions are easy to use and modify in the diverse needs that unavoidably arise. The package is also compatible with the `SOM_PAK`.

3. Applying the SOM to Industrial Systems

In this section, possibilities to use the SOM in process analysis, monitoring and modeling are discussed. Representative application examples and many references can be found in articles by Kohonen *et al.* [14] and Simula and Kangas [29].

3.1 Process Analysis and State Monitoring

In chemical process industry, for instance, the SOM can be applied to on-line observation of processes as well as off-line analysis of process data. Because clear faults seldom occur in such processes and are typically quite uninteresting, faulty situations should be filtered out from the training data set to be able to analyze normal operation more accurately.

The SOM can be used to form a display of the operational states of the process. The current process state and its history in time can be visualized as a trajectory on the map. This allows efficient tracking of the process dynamics. The SOM facilitates understanding of processes so that several variables and their interactions may be

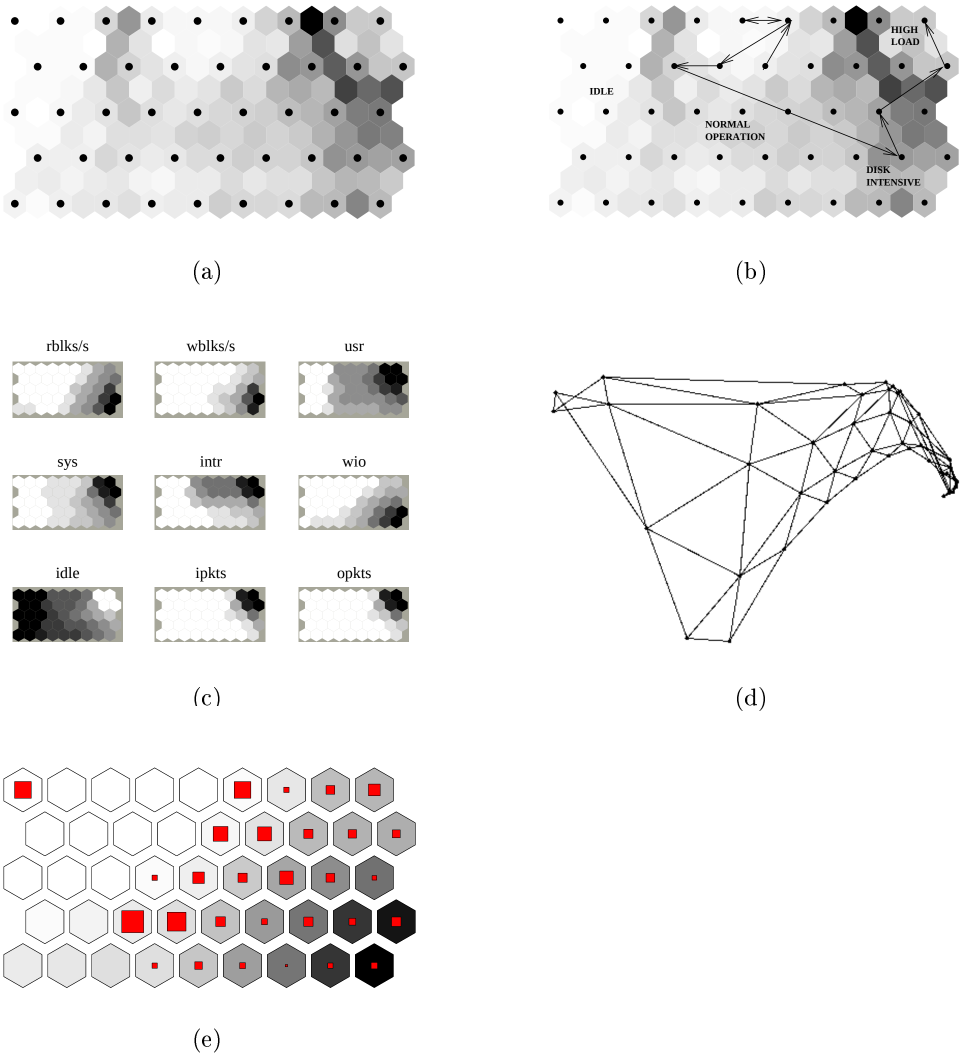


Figure 2.2. Different visualizations of the SOM. U-matrix presentation (a), trajectory on top of labeled u-matrix (b), planes representation (c), Sammon's mapping (d) and data histogram on top of a component plane (e). In figures (a) and (b) the black spots denote map units.

inspected simultaneously. Kasslin *et al.* [10] used the SOM to monitor the state of a power transformer and to indicate when the process was entering a non-desired state represented by a "forbidden" area on the map. Tryba and Goser [34] applied the SOM in monitoring of a distillation process and discussed its use in chemical process control in general.

In fault detection, the SOM should be trained using measurement vectors describing normal operation only. Thus, a mapping of the "normal operation" input space is formed. A faulty situation can be detected by monitoring the quantization error (distance between the input vector and the BMU). Large error indicates that the process is out of the normal operation space. For example, Alander *et al.* [1] and Harris [5] have used the SOM for this purpose.

The problem of fault detection and identification is more difficult. The SOM should be trained using all possible data describing both normal and abnormal situations of the process. If faulty situations are rare, measurements describing simulated faults may be added. Map units representing faulty states of the process may be labeled according to known samples. The monitoring is based on tracking of the operation point: location of the point on the map indicates the process state and the possible fault type. Vapola *et al.* [37] constructed a two-level SOM model, which was used first to detect and then to identify fault conditions in an anesthesia system.

3.2 Process Modeling

3.2.1 Regression via Local Models. General regression of y on $\boldsymbol{x}$ is usually defined by $\hat{y} = E(y|\boldsymbol{x})$. That is, the expectation of the output y given the input vector $\boldsymbol{x}$. To motivate the use of the SOM for regression, it is worth noting that the codebook vectors represent local averages of the training data.

The SOM can be used for predicting, for example, the output quality of a process given the measurements of incoming raw material characteristics and process parameter settings [6]. Regression is accomplished by searching for the BMU using the known vector components of $\boldsymbol{x}$. As an output, an approximation of the unknown components of the codebook vector are given (see Fig. 3.1).

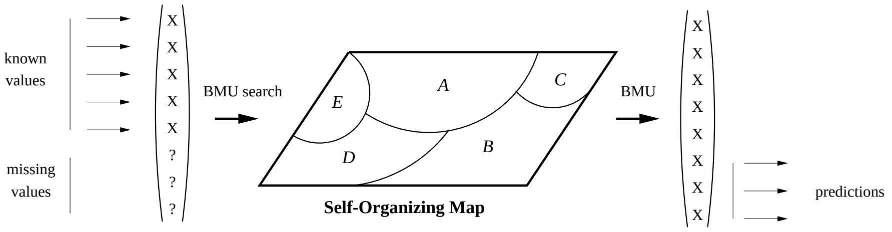


Figure 3.1. Prediction of missing components of the input vector.

The accuracy of the model can be increased by building local models for the data in the Voronoi sets of the SOM. The Voronoi set V_i of map unit i is a set of vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, for which the codebook vector $\boldsymbol{m}_i$ is closest.

The Voronoi sets provide a partitioning of the input data into disjoint sets. Each set contains points that are near each other in the data space. Subsets are modeled by independent local models, which together are considered a solution to the modeling problem. Each model is based on the local data set V_i only. This kind of approach could be called divide-and-conquer modeling. The models are not constrained to be of any specific form, or not even of similar form. In our experiments, only simple local linear models have been considered.

Total least squares type of linear regression can be performed using Principal Component Analysis (PCA) in model fitting. This approach allows measurement errors also in inputs while the usual least squares approach assumes that the input variables are accurate and there is error in the output variables only [25]. Combination of these two modeling methods takes advantage of the nonlinear elasticity of the SOM as well as the local efficiency of the PCA. Also, the topology preservation property of the SOM projection can be incorporated by allowing neighboring data sets and models to interact in some way. Ritter *et al.* [26] used this approach in local modeling of three-dimensional working space of a robot arm.

3.2.2 Sensitivity Analysis. It is often desirable to know the behavior of a system under small changes made in the system parameters. This is especially the case in industrial environments, where noise is present both in measurements and in operating conditions. In process control, the state of the process is desired to be moved to such a direction that better quality is achieved. The operation point needs to be stable: small random fluctuations in input parameters must not cause large changes in output parameters.

The model described above can be used to investigate leverage effects of small changes made in one of the process parameters. This is possible because the system cannot reach all the possible values in the space defined by the measurements, but may be limited to a low-dimensional manifold. The state space, or the space of possible values, is constrained by the characteristic behavior of the system. This is illustrated in Fig. 3.2, which depicts a two-dimensional SOM trained with data originating from a three-dimensional measurement space. As a small change in one of the measurements is imposed, the BMU changes to another map unit. By tracking the change of the BMU caused by the change of the parameters, the mutual nonlinear dependence of the parameters is revealed; one is "surfing" on a low-dimensional manifold defined by the SOM projection [6].

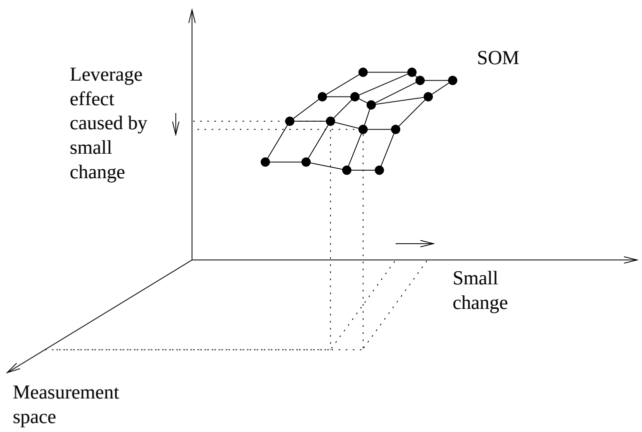


Figure 3.2. A small change along one measurement axis causes a change in other process parameters.

4. Some Case Studies

4.1 Analysis of a Continuous Pulp Digester

In a case study, a continuous pulp digester of a pulp mill was studied. Input vectors of the SOM consisted of 11 temperature sensor readings from digester sides and one output quality measurement, the kappa number. The aim of the test was to study correlation between the kappa number and the temperatures. The process temperature was beforehand known to be one of the most important factors in successful digester operation.

Input data consisted of material collected during a half year period at the mill (about 20000 measurements per channel). Erroneous measurements were filtered out from the input data using *a priori* knowledge of the process and only data depicting steady operation of the process were selected for use. The input vector components were delayed with respect to each other in time so that each vector consisted of the temperature profile of "a piece" of pulp and corresponding kappa number. After input data processing, there were 9975 12-dimensional input data vectors. A SOM of 40 by 25 units was trained using the vectors so that the BMU was searched using the 11 temperatures, and the adaptation was made using the temperatures *and* the kappa number. This way it was possible to study the correlation between temperatures and end product quality.

The component planes of the SOM are presented in Fig. 4.1. Black color indicates high temperature and white correspondingly low. For example, high temperatures in eight first measurements #1 – #8 (black spot in the middle of plane) are reflected in last plane, the kappa number, by small values (white spot in the middle). The kappa number roughly correlates (inversely) with the eight first temperature measurements. The phenomenon has clear explanation: when the cooking temperature is high, the delignification reactions are fast and the kappa number becomes low. Because the kappa number also depends on factors not included in the analysis (like concentration of cooking chemicals participating in the delignification reaction), kappa number variations that cannot be explained by temperatures are observed.

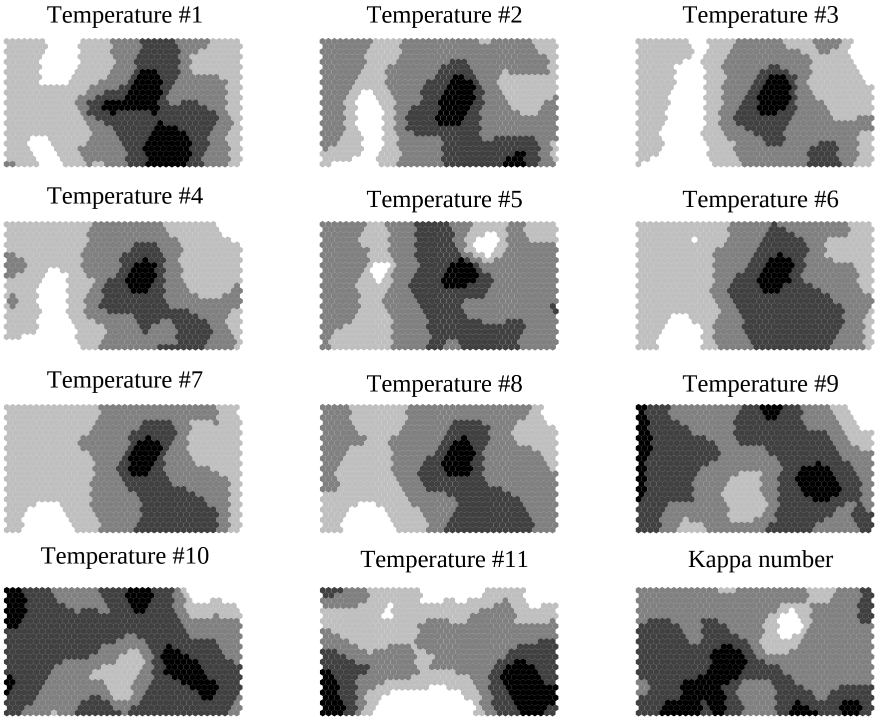


Figure 4.1. Component planes of the continuous pulp digester process.

4.2 Monitoring and Resource Allocation in Telecommunication Systems

Recently, the adaptive resource management of telecommunication networks have become more and more important with the advent of Intelligent Networks (IN), ATM, mobile communication networks etc. It is evident that the Quality of Service (QoS) requirements of these networks cannot be met by only increasing the processing capacity. Adaptive methods are needed for efficient utilization of resources. Neural networks based methods, capable for on-line monitoring of the network load, can significantly increase the efficiency of the system. Novel methods based on, e.g., the SOM algorithm have been investigated with promising results [31, 32, 33].

A Service Control Point (SCP) with the Service Data Function (SDF) in the IN is a computer system offering several kinds of services simultaenously. It is often regarded as a serious performance bottleneck of the IN. Flow control and resource management of the incoming calls are the main issues related to the performance problem. Traditionally, the IN has a distributed load control mechanism which uses an automatic call-gapping (ACG) method to limit the flow of messages in order to prevent the system from congestion. As a call rate method, ACG supports selective flow control and makes the SCP overload avoidance possible. However, it also has some disadvantages, e.g., the calculation of control parameters is sensitive to the incoming traffic due to its non-stationarity. It is difficult to find a suitable gap criteria leading to optimal SCP utilization and to determine when to activate and deactivate ACG. Therefore, the suitable gap criteria should be determined dynamically during the operation.

To find the suitable gap criteria dynamically, a good traffic prediction method for the incoming traffic should be found. Due to the non-stationarity of traffic distributions, adaptive methods may result in better throughput of the SCP and highly improved performance. The SOM is a suitable tool for monitoring the incoming call distributions, provided that the distributions are relatively slowly varying, corresponding to Poisson Calls. However, the SOM is not able to follow the fast changes which result from the so called Mass Calls, e.g., in tele-voting applications. For this purpose, a different approach must be taken.

For monitoring and adaptive decoding of the gaps, a SOM based decoding algorithm, SOM-D, has been developed [31]. The prediction of the incoming calls for different services is based on monitoring the traffic with a time interval of some minutes, e.g. 10-15 minutes. This is satisfactory for Poisson Call distributions. The resources are then assigned to different services for the next time interval according to this monitoring. The experimental tests with SOM-D using both simulated and true traffic data have shown improvements from 20 % up to 70 % when compared to the ACG with the fixed gap criteria.

The incoming calls in Mass Call arrival process are rapidly changing and they usually have a burst-like distribution. Therefore, the time interval of 10–15 minutes in SOM-D adaptation is far too long. A good prediction method and a suitable ACG management system with time interval of only a few seconds is required. Therefore, a feed-forward neural network (FFN) is used in parallel with the SOM-D algorithm to adapt to Mass Calls. So, this adaptation mechanism, called FFNSOM-D, is an extended version of the SOM-D algorithm. It is capable of controlling both Poisson Call and Mass Call distributions. The network predicts the Mass Call distribution in a short time interval, e.g. between 5 to 10 seconds. The prediction is based on modeling the Mass Call process using a Markov Modulated Poisson Process. The one-step-ahead traffic prediction for Mass Calls, is completed with a simple feed-forward neural network which is trained beforehand. The FFNSOM-D algorithm is described in detail in [33].

The model of a multi-service SCP with the distributed call gapping mechanism utilizing FFNSOM-D is depicted in Fig. 4.2. The gaps are adaptively determined for the calls corresponding to the different services. SOM-D receives the cumulative traffic and adapts in the long time interval (10–15 minutes), while FFN and the boundary function adapt in the short time interval (5–10 seconds) based on traffic modeling. Extensive simulations have shown that the FFNSOM-D method is superior when compared with the plain ACG method. The throughput increases linearly with the incoming calls and reaches the upper limit of the system throughput.

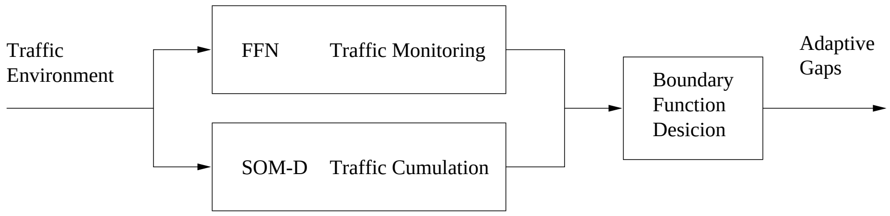


Figure 4.2. FFNSOM-D call gapping mechanism.

4.3 Modeling of a Steel Process

The methods described in Sect. 3.2 were used in modeling of a steel production process. The part of the process studied consists of a cold rolling and batch annealing process.

Prior to cold rolling, steel coils are opened and pickled in an acid bath to remove corrosion and excess dirt from the surface of the coil. In cold rolling, the coils are rolled using a tandem mill, where a few consecutive rolls reduce the thickness of the coil. In addition to thinning effects, this improves the strength of the steel. Then, a batch annealing follows the rolling operation. The coils are heated to high temperature, after which they are cooled off slowly. This improves the microstructure of the steel which affects such qualities as strength and elasticity. As the last processing step, the coils go through a temper mill, where a slight reduction in the thickness is made. This improves the microstructure of the steel as well as the strength.

The analysis was based on the measurements made in the production environment. The inputted steel coils are characterized by their element concentrations (C Si Mn P S Al Nb V Ti Cu Cr Ni Mo B N), the production environment by the pre-set reductions of the steel rolls and the end quality, which is measured by three quality parameters that reflect the mechanical properties of steel (such as tensile strength). The goal was to predict the end product quality with the knowledge of inputted steel element concentrations as well as production parameters. In this way, costly off-line measurements could be replaced by on-line prediction of the output quality.

The measurements were collected from the factory automation system and divided to two different sets: the training set (2306 coils) and the testing set (906 coils). The models were trained with the training set and their performance tested with the testing set not used in training. When testing the model, the measured end product quality was compared with model output to produce an error estimate of the models (see Table 4.1). The first row of the Table demonstrates the inferiority of linear models in this application. The next three rows show the performance of the SOM models at three different resolutions. It can be clearly seen that increasing the number of codebook vectors improves the accuracy. By the introduction of local linear models, we can further improve the accuracy of these models.

Table 4.1. Prediction errors for an independent testing set.

Method	MSE
Global PCA	1.8858
SOM (8×6 units)	0.8098
SOM (14×10 units)	0.6059
SOM (20×14 units)	0.5668
SOM (8×6 units) and local PCA	0.6785

4.4 Time-Series Prediction

In the last decade, local models have been a source of much interest in modeling because in many cases they give better results than global models [30]. Local models are usually based on nearest-neighbour methods. A good set of prototype vectors for this purpose can be acquired by quantizing the feature space. The local models themselves can be constructed in various ways. Usually, they are simple such as weighted averages of the prototype vectors, or linear regression models.

The SOM is an efficient method for vector quantization. Previously, it has been used in chaotic time-series prediction e.g. by Walter *et al.* [40], Der and Herrman [4], Principe [24] and Vesanto [38]. Typically, the local models are constructed simultaneously with the SOM using some updating procedure. The local models can also be estimated after the SOM has been trained. A generalized scheme for constructing local models in this way is depicted in Fig. 4.3. The data are embedded and divided into training and testing sets. Based on the training set, the embedded data space is quantized. Then, local data sets are constructed for each prototype vector. The local data set of a prototype vector consists of data vectors for which it is closest (i.e. of data vectors belonging to the prototype vector's Voronoi plate). If a local data set of a prototype vector is too small, it can be augmented using the data sets of similar prototype vectors lying close to each other. At the last step, local models are constructed based on the local data sets.

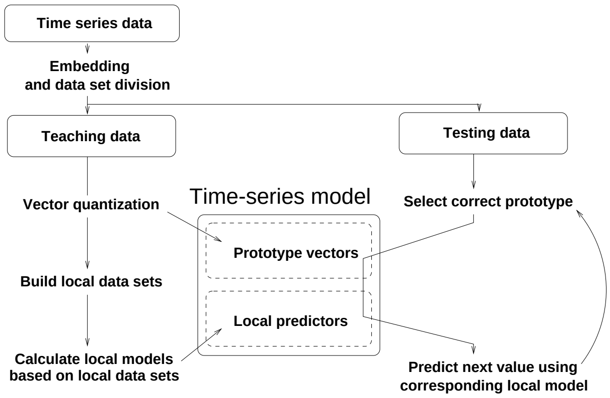


Figure 4.3. A generalized scheme for constructing and testing the local models.

The time-series $x(n)$ used in testing the method was the Mackey-Glass system [17] (with time delay parameter equal to 17), which is a widely used benchmark data set.

Separate and independent training and testing sets were generated. The embedded data vectors consisted of four values of the time-series: $\boldsymbol{x}(n) = [x(n), x(n-6), x(n-12), x(n-18)]^{\mathrm{T}}$. This feature vector was used to predict $x(n+6)$. The predicted value was inserted into the feature vector and this process was iterated 14 times so that the final value corresponded to $x(n+84)$.

The performance of the proposed method was first tested on various sized SOMs and data sets of different size. Also other quantization methods were tested: the k-means and the neural-gas [18] algorithms.

In Table 4.2, the results are compared with those of other methods. It should be noted that the training set size affected the results considerably. Therefore, only results with equal training sets should be compared. Deco and Obradovic used decorrelated Hebbian learning algorithm to predict the time-series at $x(n+85)$ [3]. Platt used a resource allocating network [23] and Littman and Ritter a direct cascade architecture with local linear map subnetworks in the same task [16]. Lapedes and Farber used standard backpropagation network with 500 training samples [15].

Table 4.2. Normalized RMS errors produced by various methods in predicting the values either $x(n+84)$ or $x(n+85)$ of the Mackey-Glass time-series.

Method	Units	Training set size	NRMS at $x(n+84/85)$
Backpropagation [15]	n/a	500	0.06
Direct Cascade Architecture [16]	n/a	500	0.043
Decorrelated Hebbian Learning [3]	n/a	1000	0.056
SOM (35 × 35)	1225	1000	0.035
Neural gas (500 units)	500	3000	0.065
Neural gas (1000 units)	1000	3000	0.062
SOM (10 × 10)	100	3000	0.060
Resource Allocating Network [23]	n/a	3000	0.054
k-means (500 means)	500	3000	0.029
k-means (1000 means)	1000	3000	0.024
SOM (35 × 35)	1225	3000	0.022

The proposed method compares well with these results. With training set of 1000 samples and $35 \times 35 = 1225$ prototype vectors the proposed method produced normalized RMS (NRMS) errors of 0.035 for $x(n+84)$. With only 100 prototype vectors, but with a larger data set (3000 points) a slightly weaker result, 0.060, was achieved. Error of 0.022 was achieved with 3000 training samples and a map of 35 by 35 units.

4.5 Data Mining of the World's Pulp and Paper Mills Data

In this study, the SOM was used for data mining to analyze the technology data of the world's pulp and paper industry. There were three data sets containing information of (1) production capacities of different product types in pulp and paper mills, (2) technology of paper machines and (3) technology of pulp lines. Since each mill could contain several paper machines and pulp lines, a hierarchical structure of maps was used, illustrated in Fig. 4.4. The two low-level maps extracted relevant information regarding the paper machines and pulp lines data of a mill and the high-level map combined this with production capacity data.

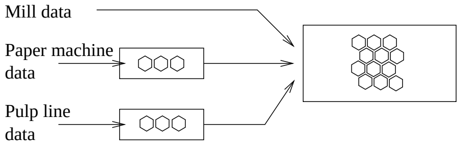


Figure 4.4. The hierarchical map structure. Data histograms from individual low level maps were utilized in training the combined high level map.

The low level maps (size 20 by 1 units) were constructed using paper machine and pulp line data sets. The seven component planes of the paper machine map are shown in Fig. 4.5. Based on the map, three main types of paper machines could be determined: (1) new paper machines with very big capacity, (2) paper machines with big paper weight and (3) paper machines whose size and capacity decrease steadily with increasing age. A similar study of the pulp line map produced three main pulp line types: (1) those using waste paper for pulp, (2) those making unbleached pulp and (3) those making chemical pulp mainly from wood fiber.

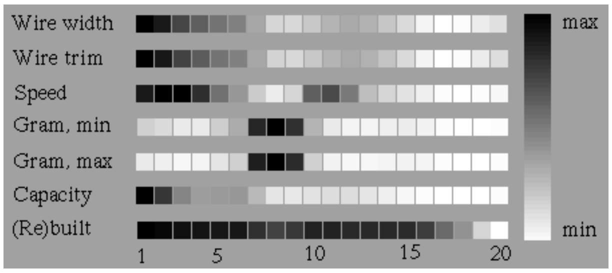
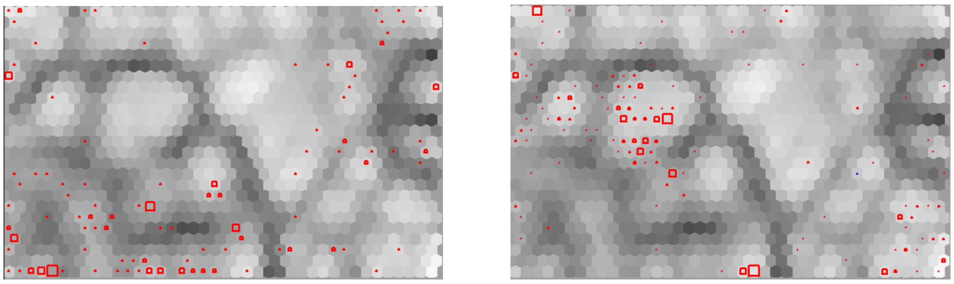


Figure 4.5. The component planes of the paper machine map: the component names on the left and the corresponding values in the map units in the middle. The three first components have a strong correlation, as do the next two components.

The high level map (size 40 by 25 units) was trained using mill-specific production capacity information of the first data set and histograms from the low level maps. For each mill, the histograms were computed using data of the paper machines and pulp lines of the mill. Figs. 4.6a and 4.6b show the u-matrix of the high level map, and the distribution of Scandinavian and Chinese pulp and paper mills on the map, respectively. The two sets are easily separable although there was no geographic information present in the data.

Scandinavia represents a technologically advanced region. The mills are new, they have high-capacity paper machines and the majority produces printing/writing paper or pulp. Chinese mills, on the other hand, have many machines and they produce both industrial and printing/writing paper. It can also be seen that in the area where the majority of the Chinese mills are the values of the u-matrix are very low. That is, the variation between weight vectors in that area is small, which means that these mills resemble each other.



(a) Scandinavian pulp and paper mills

(b) Chinese pulp and paper mills

Figure 4.6. The data set histograms of two different geographical regions on the u-matrix of the high level map. The bigger the square, the more mills were projected to that unit on the map.

5. Conclusions

The use of the SOM in analysis, monitoring and modeling of various systems has been discussed. The SOM is especially suitable in tasks which require processing of large amounts of numerical data. The method is readily explainable, simple and highly visual.

The SOM provides data-driven approach to process monitoring. When using the SOM, it is not necessary to define the process model analytically. The SOM has the desirable feature of describing nonlinear relationships between a large number of parameters and variables of complex systems phenomenologically. By using a history of measurements, dynamical behavior of the process can be introduced into the map, or a set of maps. This approach has been used to model the sequence of states and based on that to predict the future state in the system operation [24, 28].

The SOM facilitates visual understanding of processes. For instance, process operation personnel may learn to adjust the control variables in such a way that the operation point, or trajectory, stays in the desired region on the map. In this way, correct control action may be easily learned based on the visual output [21].

The construction of feature vectors allows fusion of different data and measurement sources. Information from various databases can be integrated, e.g., using SOMs in a hierarchical way. For instance, in the analysis of complex industrial processes, technical, economical, and environmental data can be combined [39]. This allows the analysis and simulation of various effects in the entire field of industry. For instance, the influence of various technical investments can be analyzed in the factory level.

Acknowledgement. This work has been carried out in the technology program "Adaptive and Intelligent Systems Applications" financed by the Technology Development Center of Finland (TEKES). The cooperation of Rautaruukki Strip Products, Rautaruukki Research Center, Jaakko Pöyry Consulting and UPM-Kymmene is gratefully acknowledged.

References

1. J. T. Alander, M. Frisk, L. Holmström, A. Hämäläinen, and J. Tuominen. Process error detection using self-organizing feature maps. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume II, pages 1229–1232, Amsterdam, Netherlands, 1991. North-Holland.
2. E. Alhoniemi, J. Himberg, K. Kiviluoto, J. Parviainen, and J. Vesanto. SOM Toolbox for Matlab. Available via WWW at *http://www.cis.hut.fi/projects/somtoolbox/*, 1997.
3. G. Deco and D. Obradovic. Decorrelated hebbian learning for clustering and function approximation. *Neural Computation*, 7:338–348, 1995.
4. R. Der and M. Herrmann. Nonlinear chaos control by neural nets. In *Proc. of ICANN'94*, pages 1227–1230, 1994.
5. T. Harris. A Kohonen S.O.M. based, machine health monitoring system which enables diagnosis of faults not seen in the training set. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN'93), Nagoya, Japan*, volume I, pages 947–950. IEEE Service Center, 1993.
6. J. Hollmén and O. Simula. Prediction models and sensitivity analysis of industrial production process parameters by using the self-organizing map. In *Proc. of IEEE Nordic Signal Processing Symposium (NORSIG'96)*, pages 79–82, 1996.
7. J. Iivarinen, T. Kohonen, J. Kangas, and S. Kaski. Visualizing the clusters on the self-organizing map. In C. Carlsson, T. Järvi, and T. Reponen, editors, *Proc. Conf. on Artificial Intelligence Res. in Finland*, pages 122–126, 1994.
8. S. Kaski. Data exploration using self-organizing maps. *Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82*, 1997. DTech Thesis, Helsinki University of Technology, Finland.
9. S. Kaski and K. Lagus. Comparing self-organizing maps. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of ICANN96, International Conference on Artificial Neural Networks, Bochum, Germany, July 16-19*, Lecture Notes in Computer Science, vol. 1112, pages 809–814. Springer, Berlin, 1996.
10. M. Kasslin, J. Kangas, and O. Simula. Process state monitoring using self-organizing maps. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume II, pages 1531–1534, Amsterdam, Netherlands, 1992. North-Holland.

11. K. Kiviluoto. Topology preservation in self-organizing maps. In *Proc. of Int. Conf. on Neural Networks (ICNN'96)*, volume 1, pages 294 – 299, New York, NY, USA, 1996. IEEE.
12. T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences.* Springer, Berlin, Heidelberg, 1995.
13. T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK: The Self-Organizing Map Program Package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996. Available via WWW at *http://www.cis.hut.fi/nnrc/*.
14. T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358 – 1384, 1996.
15. A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report TR LA-UP-87-2662, Los Alamos National Laboratory, Los Alamos, 1987.
16. E. Littman and H. Ritter. Learning and generalization in cascade network architectures. *Neural Computation*, 8:1521–1539, 1996.
17. M. C. Mackey and L. Glass. Oscillations and chaos in physiological control systems. *Science*, 197:287, 1977.
18. T. M. Martinetz, S. G. Barkovich, and K. J. Schulten. "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, July 1993.
19. Mathworks Inc. *Using Matlab Version 5*, 1996. (Mathworks WWW address: *http://www.mathworks.com/*).
20. J. W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.
21. R. Otte and K. Goser. New approaches of process visualization and analysis in power plants. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 44–50. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.
22. W. Pedrycz and H. C. Card. Linguistic interpretation of self-organizing maps. In *Proc. of International Conference on Fuzzy Systems '92*, 1992.
23. J. Platt. Learning by combining memorization and gradient descent. *Advances in Neural Information Processing Systems*, 1991.
24. J. C. Principe and L. Wang. Non-linear time series modeling with Self-Organization Feature Maps. In *Proc. NNSP'95, IEEE Workshop on Neural Networks for Signal Processing*, pages 11–20. IEEE Service Center, 1995.
25. C. R. Rao and H. Toutenburg. *Linear models: least squares and alternatives.* Springer-Verlag, New York, 1995.
26. H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps.* Addison-Wesley Publishing Company, 1992.
27. T. Samad and S. A. Harp. Self-organization with partial data. *Network: Computation in Neural Systems*, 3(2):205–212, 1992.
28. O. Simula, E. Alhoniemi, J. Hollmén, and J. Vesanto. Monitoring and modeling of complex processes using hierarchical self-organizing maps. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'96)*, volume Supplement, pages 73–76, 1996.
29. O. Simula and J. Kangas. *Neural Networks for Chemical Engineers*, volume 6 of *Computer-Aided Chemical Engineering*, chapter 14, Process monitoring and visualization using self-organizing maps. Elsevier, Amsterdam, 1995.
30. A. C. Singer, G. W. Wornell, and A. V. Oppenheim. Codebook prediction: A nonlinear signal modeling paradigm. In *Proc. ICASSP'92*, pages 325–328, 1992.
31. H. Tang and O. Simula. Neural adaptation for optimal traffic shaping in telephone systems. In *Proc. ICNN'95, IEEE Int. Conf. on Neural Networks*, volume IV, pages 1561–1565, Piscataway, NJ, 1995. IEEE Service Center.
32. H. Tang and O. Simula. The optimal utilization of multi-service scp. In *Proc. of IFIP-TCG Working Conf. on Intell. Networks, Copenhagen, Denmark*, pages 157–167, 1995.
33. H. Tang and O. Simula. The adaptive resource assignment and optimal utilization of multi-service scp. In *Proceedings of the 4th International Conference on Intelligence in Networks (ICIN'96), Bordeaux, France*, pages 235–240, 1996.
34. V. Tryba and K. Goser. Self-Organizing Feature Maps for process control in chemistry. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 847–852, Amsterdam, Netherlands, 1991. North-Holland.

35. V. Tryba, S. Metzen, and K. Goser. Designing basic integrated circuits by self-organizing feature maps. In *Neuro-Nimes '89. International Workshop. Neural Networks and their Applications*, pages 225 – 235, 1989.
36. A. Ultsch and H. Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In *Proc. INNC'90, Int. Neural Network Conf.*, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer.
37. M. Vapola, O. Simula, T. Kohonen, and P. Meriläinen. Representation and identification of fault conditions of an anaesthesia system by means of the Self-Organizing Map. In M. Marinaro and P. G. Morasso, editors, *Proc. ICANN'94, Int. Conf. on Artificial Neural Networks*, volume I, pages 350–353, London, UK, 1994. Springer.
38. J. Vesanto. Using the som and local models in time-series prediction. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 209–214. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.
39. J. Vesanto, P. Vasara, R.-R. Helminen, and O. Simula. Integrating environmental, technologigal and financial data in forest industry analysis. In *Proceedings of SNN'97 Europe's Best Neural Networks Practice, Amsterdam, Netherlands*, 1997. (to appear).
40. J. Walter, H. Ritter, and K. Schulten. Non-linear prediction with self-organizing maps. In *Proc. IJCNN-90-San Diego, Int. Joint Conf. on Neural Networks*, volume 1, pages 589–594. IEEE Service Center, Piscataway NJ, 1990.