Time Series Prediction using Recurrent SOM with Local Linear Models

Timo Koskela, Markus Varsta, Jukka Heikkonen, and Kimmo Kaski
Helsinki University of Technology
Laboratory of Computational Engineering
P.O. Box 9400, FIN-02015 HUT, FINLAND
(Invited Paper)

Keywords: Time Series Prediction, Local Models, Recurrent Self-Organizing Map

Abstract

A newly proposed Recurrent Self-Organizing Map (RSOM) is studied in time series prediction. In this approach RSOM is used to cluster the data to local data sets and local linear models corresponding to each of the map units are then estimated based on the local data sets. A traditional way of clustering the data is to use a windowing technique to split it to input vectors of certain length. In this procedure, the temporal context between the consecutive vectors is lost. In RSOM the map units keep track of the past input vectors with a recurrent difference vector in each unit. The recurrent structure allows the map to store information concerning the change in the magnitude and direction of the input vector. RSOM can thus be used to cluster the temporal context in the time series. This allows a different local model to be selected based on the context and the current input vector of the model. The studied cases show promising results.

I. Introduction

The main motivation for the time series research is the desire to predict the future and to understand the underlying phenomena and processes of the system under study. Time series prediction, also called temporal sequence processing (TSP), concentrates on building models of the process using the knowledge and information that is available. The constructed model can then be used to simulate the future events of the process.

A time series consist of measurements or observations of a natural, technical or economic process, that are made sequentially in time. In general, consecutive samples of a time series are dependent on each other to an extent dictated by the process in question. Because of this dependency, it makes sense and it is possible to predict the future of the series. Whether the dependency is linear or nonlinear in nature, it can be estimated from the measured series using statistical methods. However, measurements usually contain noise that cannot be cancelled out. Another problem is that the outcome of the process may be dependent on some unknown variables, whose effect on the process cannot be estimated. The problem is now to construct a model that can capture the essential features of the process using the measurements that are available. The model that can predict the future of a time series most accurately is usually considered to characterize the process best.

In science, technology and economy there exist many interesting processes and phenomena, whose prediction is either very useful or profitable. These include different industrial processes that can be modeled, predicted and controlled based on some sensory data. Many phenomena of nature, such as daily rainfall or the probability of an earthquake would also be useful to predict. Medical applications include for example modeling biological signals such as EEG or ECG to understand better the patients state. In economy stock market prices are an example of a possibly very profitable prediction task.

It should be pointed out, however, that there exists the question whether a process is predictable or not. In nature and in economy one can find complex processes that show self-organizing criticality [1]. In this case the answer to the question is not obvious. We have, however, passed by this question by taking here the view that a process is somehow predictable or in fact modelable—accurately or not—as long as there is statistical data available.

A time series can be nonstationary but still predictable if the changes in the statistics of the process are tractable [20]. In this case, model parameters can be changed adaptively to reflect these changes. Adaptation of the parameters can be carried out i.e. with Kalman filter [11]. Another case is when the changes in the statistics of the process can be explained by different states in the process. In this case, it is possible to identify local models for each of the states. The prediction of the time series is then produced by predicting first the state of the process, and then using the corresponding local model to gain the desired prediction.

Various approaches to time series prediction have been studied over the years [7]. Among linear regression methods autoregressive (AR) and autoregressive moving average (ARMA) [2] models have been the most used methods in practice [24] since Yule's paper in the 1920's [30]. The theory of linear models is well known, and many algorithms for model building are available.

Multivariate adaptive regression splines (MARS) [6] is an example of a more complex regression method. Model identification is carried out by over-fitting the data first with nonparametric splines. In second phase the parameters of a linear sum of the splines is estimated and the whole process is continued until convergence.

In practice almost all measured processes are nonlinear to some extent and hence linear modeling methods turn out to be in some cases inadequate. Nonlinear methods became widely applicable in 1980's with the growth of computer processing speed and data storage. Of the nonlinear methods neural networks soon became very popular. Neural networks are biologically inspired models that typically consist of simple computing units connected in some manner, see e.g. [8]. Structure of the connections between units and the calculation that a unit performs vary in different neural models. Characteristic for neural networks is that the optimization of all the model parameters is carried out at the same time with a learning algorithm. Many different types of neural networks, such as multilayer perceptron (MLP) and radial basis function network (RBF) have been proven to be universal function approximators [5]. This gives neural networks certain attractiveness in time series modeling, see e.g. [16], [19] and [25].

Different models can be divided into global and local models. In global model approach only one model is used to characterize the measured process. Traditional models, such as autoregressive models (AR) and multilayer perceptron (MLP) when used as in [15] are examples of global models. Global models give best results with stationary time series. However, when the series is nonstationary, identifying a proper global model becomes more difficult. During the last decade, local models have gained a growing interest, because they can often overcome some of the problems of the global models [23]. Local models are based on dividing the data set to smaller sets of data, each being modeled with a simple local model.

An example of an extension of the AR model to build a local linear model is the threshold autoregressive (TAR) [24] model. TAR consist of separate AR models that are used in different parts of the data according to threshold values. The selection of the current AR model used for prediction is carried out by comparing the value of the time series at certain point in the past with the threshold values. The model selection problem of the TAR model contains deciding the number of local data sets, the number of parameters in each model, and the time lag with which the model selection is carried out. The estimation of the model includes searching for the threshold values and the parameters of local AR models.

In the local model approach division of the data set is usually carried out with some clustering or quantization algorithm such as k-means, Self-Organizing Map (SOM) [28], [27] or neural gas [18]. Another approach using self-organizing capability and local models is presented in [14]. After clustering the data, local models for the generated local data sets are estimated. The problem of data division for achieving the best prediction accuracy remains in a general case to be solved. One method using an iterative learning algorithm to solve the division of the data and the local model parameters at the same time is Hierarchical Mixtures of Experts [10]. Here a gating network decides which local model is to be used in prediction. The parameters of local models and the gating network are optimized with iterative Expectation-Maximization algorithm.

Selection of the input to the model is usually done by using a windowing technique to split the time series into input vectors. The problem is thus converted into selection of the type and length of the window used. Typically the input vectors contain past samples of the series up to certain length. The time is in effect converted into additional spatial dimension of the input vectors and the temporal context between consecutive samples is lost.

When the series is stationary with certain time lag, splitting can be carried out without losing too much from the original information. When the series is nonstationary, it can be difficult to determine a proper window length. This is a considerable problem for global nonadaptive models. If the window is too long, it tends to average out the changes in the statistics of the process. If the window is too short, is is sensitive to short disturbances in the series. In general, the optimal window length for the model varies in the time series, and thus the windowing should be adaptive. This approach, however, can be used effectively only when the changes in the statistics of the process are slow.

Another approach is to try to capture the essential context that is lost in windowing with some other techniques. One way is to store the contextual information that exists between the samples and consecutive vectors to a memory in the model. The memory can store relevant information from the past, which is not visible in current input vector of the model. For instance, memory can store the current state of the process, which can then be used in selection of the local model used for prediction.

Our approach in this study is to use Recurrent Self-Organizing Map (RSOM) [26] to store temporal context from the input vectors. The map consists of units that are each associated with a local linear model. RSOM is used to cluster the time series into local data sets, which are then used in the estimation of the corresponding local linear models.

The rest of the paper is organized as follows: In the second section newly proposed Recurrent SOM (RSOM) architecture and learning algorithm is introduced. Using RSOM for building the local data sets from the time series is presented. In the third section different prediction cases are studied. Implementation of RSOM with linear regression methods for prediction and methods used for model selection and evaluation are presented. In three cases of different time series results of RSOM are compared with linear and nonlinear global models (AR and MLP). Finally some conclusions are made.

II. Modeling Temporal Context with Recurrent Self-Organizing Map

As discussed earlier, the temporal context in the series can be stored in the model by using some kind of memory structure. We present as an extension to the Self-Organizing Map the Recurrent Self-Organizing Map (RSOM) that allows storing certain information from the past input vectors. The information is stored in the form of difference vectors in the map units. The mapping that is formed during training has the topology preservation characteristic of the SOM as explained in the next section.

A Recurrent SOM can be used to determine the context of the consecutive input vectors by performing what we call "temporal quantization". It is now possible to use the context with the current input of the model to select a different local model in cases where the current input vector looks the same but the context is different. In prediction the closest reference vector of the RSOM, called the best matching unit, is searched for each input vector. A local model that is associated with the best matching unit is then selected to be used in the prediction task at that time.

Next we will present the theory of the Recurrent Self-Organizing Map and describe how temporal quantization is implemented with it.

A. Recurrent Self-Organizing Map

The Self-Organizing Map (SOM) [13] is a vector quantization method with topology preservation when the dimension of the map matches the true dimension of the input space. In brief topology preservation means that input patterns close in the input space are mapped to units close on the SOM lattice, where the units are organized into a regular N-dimensional grid. Furthermore units close on the SOM are close in the input space. Topology preservation is achieved with the introduction of topological neighborhood that connects the units of the SOM with a neighborhood function, h.

The training algorithm of the SOM is based on unsupervised learning, where one sample, the input vector $x(n)$ from the input space V_I, is selected randomly and compared against the weight vector w_i of the unit i the map space V_M. The best matching unit b to given input pattern $x(n)$ is selected using some metric based criterion, such as

$$\|x(n) - w_b(n)\| = \min_{i \in V_M} \{\|x - w_i(n)\|\} , \qquad (1)$$

where the parallel vertical bars denote the Euclidean vector norm. Initially all weight vectors are set randomly to their initial positions in the input space. During the learning phase the weights in the map are updated towards the given input pattern $x(n)$ according to

$$w_i(n+1) = w_i(n) + \gamma(n) h_{ib}(n)(x(n) - w_i(n)) , \qquad (2)$$

where $i \in V_M$ and $\gamma(n)$, $0 \leq \gamma(n) \leq 1$, is a scalar valued adaptation gain. The neighborhood function, $h_{ib}(n)$, gives the excitation of unit i when the best matching unit is b. A typical choice for h_{ib} is a Gaussian function $h_{ib}(n) = exp(-\|r_i - r_b\|^2/\sigma(n)^2)$, where σ controls the width of the function and r_i, r_b are the N-dimensional SOM index vectors of the unit i and the best matching unit b. During learning the function h_{ib} normally approaches delta function, i.e., σ slowly approaches zero as training progresses. When good quantization is desired the map has to be trained with only b in h_{ib} once the map has organized. During this quantization stage the gain has to be sufficiently small to avoid losing the map order, how small exactly varies from case to case.

In order to guarantee convergence of the algorithm, the gain $\gamma(n)$ should decrease as a function of time or training steps according to conditions [22]:

$$\begin{aligned}\lim_{t\to\infty}\int_0^t \gamma(t')dt' &= \infty \;, \\ \lim_{t\to\infty}\int_0^t (\gamma(t'))^2 dt' &= C,\; C<\infty \;. \end{aligned} \quad (3)$$

If the map is trained properly, i.e the gain and the neighborhood functions are properly decreased over training a mapping is formed, where weight vectors specify centers of clusters satisfying the vector quantization criterion:

$$E = min\{\sum_{j=1}^{M} ||x_j - w_{b(x_j)}||\}, \quad (4)$$

where we seek to minimize the sum squared distance E of all input patterns, $x_j, j = 1 \ldots M$, to the respective best matching units with weight vectors $w_{b(x_j)}$.

Furthermore [4] relates the point density function, $p(w)$, of the weight vectors with the point density function, $p(x)$, of the sampled underlying distribution in the V_I. Density function $p(w)$ is a better approximation of the underlying $p(x)$ than the approximation $p(x)^{\frac{d}{d+2}}$. Here d is the dimension of the V_I, achieved with a random quantizer.

Chappell and Taylor [3] proposed a modification to the original SOM. This modification, being named Temporal Kohonen Map, is not only capable of separating different input patterns but is also capable of giving context to patterns appearing in sequences. Next we will describe Temporal Kohonen Map in more detail.

A.1 Temporal Kohonen Map

Temporal Kohonen Map (TKM) differs from the the SOM only in its outputs. The outputs of the normal SOM are reset to zero after presenting each input pattern and selecting the best matching unit with the typical winner take all strategy or making other use of the unit output values, hence the map is sensitive only to the last input pattern. In the TKM the sharp outputs are replaced with leaky integrator outputs which, once activated, gradually lose their activity.

The modeling of the outputs in the TKM is close to the behavior of natural neurons, which retain an electrical potential on their membranes with decay. In the TKM this decay is modeled with the difference equation:

$$V_i(n) = dV_i(n-1) - (1/2)\|x(n) - w_i(n)\|^2 \;, \quad (5)$$

where $0 < d < 1$ can be viewed as a time constant, $V_i(n)$ is the activation of the unit i at step n, $w_i(n)$ is the reference or the weight vector in the unit i and $x(n)$ is the input pattern. Now the best matching unit b is the unit with maximum activity. Equation Eq. 5 has the following general solution:

$$V_i(n) = -(1/2)\sum_{k=0}^{n-1} d^k \|x(n-k) - w_i(n-k)\|^2 + d^n V_i(0) \;, \quad (6)$$

where the involvement of the earlier inputs is explicit. Further analysis of Eq. 6 shows how the optimal weight vectors in the vector quantization sense can be solved explicitly when n is assumed to be sufficiently large to render the last residual term corresponding to initial activity insignificant. The analysis, when w_i is assumed constant, goes as follows:

$$\frac{\partial V(n)}{\partial w} = -\sum_{k=0}^{n-1} d^k (x(n-k) - w). \quad (7)$$

Now, when w is optimal in the vector quantization sense (Eq. 4) the derivative in Eq. 7 is zero as this minimizes the sum in Eq. 6. Hence substituting the left hand side of Eq. 7 with 0 yields:

$$\begin{aligned} 0 &= -\sum_{k=0}^{n-1} d^k (x(k) - w) \;, \\ w &= \frac{\sum_{k=0}^{n-1} d^k (x(n-k))}{\sum_{k=0}^{n-1} d^k}. \end{aligned} \quad (8)$$

The result shows how the optimal weight vectors in the vector quantization sense are linear combinations of the input patterns. Since the TKM is trained with the normal SOM training rule, it attempts to minimize the normal vector quantization criterion in Eq. 4, which is other than the criterion suggested by Eq. 8. As a consequence it appears that it may be possible to properly train a TKM only for relatively simple input spaces.

A.2 Modified TKM: RSOM

Some of the problems of the original TKM have a convenient solution in simply moving the leaky integrators from the unit outputs into the inputs. This gives rise to the modified TKM called Recurrent Self-Organizing Map, RSOM.

Moving the leaky integrators from the outputs into the inputs yields

$$y_i(n) = (1-\alpha)y_i(n-1) + \alpha(x(n) - w_i(n)) \;, \quad (9)$$

for the temporally leaked difference vector at each map unit. Above $0 < \alpha \leq 1$ is the leaking coefficient analogous to d in the TKM, $y_i(n)$ is the leaked difference vector while $x(n)$ and $w_i(n)$ have

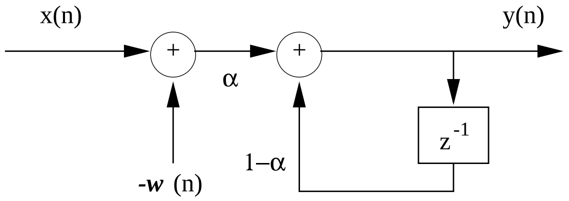

Fig. 1. Schematic picture of an RSOM unit which acts as a recurrent filter.

their previous meanings. Schematic picture of an RSOM unit is shown in Fig. 1.

Large α corresponds to short memory while small values of α correspond to long memory and slow decay of activation. In the extremes of α RSOM behaves like a normal SOM ($\alpha = 1$) while in the other extreme all units tend to the mean of the input data.

Eq. 9 can be written in a familiar form by replacing $x_i'(n) = x(n) - w_i(n)$, yielding:

$$y_i(n) = (1-\alpha)y_i(n-1) + \alpha x_i'(n); \tag{10}$$

which describes an exponentially weighted linear IIR filter with the impulse response $h(k) = \alpha(1-\alpha)^k$, $k \geq 0$. For further analysis of the Eq. 10, see e.g. [21].

Since the feedback quantity in RSOM is a vector instead of a scalar it also captures the direction of the error which can be exploited in weight update when training the map. The best matching unit b at step n is now searched by

$$y_b = min_i\{||y_i(n)||\}\ , \tag{11}$$

where $i \in V_M$. Then the map is trained with a slightly modified Hebbian training rule given in Eq. 2 where the difference vector, $(x(n) - w_i(n))$ is replaced with y_i. Thus the unit is moved toward the linear combination of the sequence of input patterns captured in y_i.

Repeating the mathematical analysis on RSOM earlier done with the TKM in Eqs. 7 and 8 yields:

$$y(n) = \alpha \sum_{k=1}^{n} (1-\alpha)^{(n-k)}(x(k) - w).$$

The square of the norm of $y(n)$ is

$$||y(n)||^2 = \alpha \sum_{k=1}^{n} (1-\alpha)^{(n-k)}(x(k)-w)^T(x(k)-w). \tag{12}$$

Optimizing the vector quantization criterion given in general form in Eq. 4 with respect to y yields the following condition:

$$\frac{\partial ||y(n)||^2}{\partial w} = -2\alpha \sum_{k=1}^{n} (1-\alpha)^{(n-k)}(x(k) - w) = 0, \tag{13}$$

when w is optimal. The optimal w can be analytically solved:

$$w = \frac{\sum_{k=1}^{n} (1-\alpha)^{(n-k)}\ x(k)}{\sum_{k=1}^{n} (1-\alpha)^k}. \tag{14}$$

From Eq. 14 one immediately observes how the optimal w's are linear combinations of the x's. Note how this result is essentially identical with the result in Eq. 8 so it might seem that the algorithms are essentially the same. However since RSOM is trained with the y's it seeks to minimize the quantization criterion suggested by Eq. 13 while the TKM seeks to minimize the normal vector quantization criterion in Eq. 4. Nevertheless the resolution of RSOM is limited to the linear combinations of the input patterns with different responses to the operator in the unit inputs. If a more sophisticated memory is required one has to resort to multilayer structures like those presented in [12].

B. Implementing Temporal Quantization

In RSOM the feedback in the difference vectors allows the past to affect the mapping that is formed. The best matching unit at each time is selected according to the (Euclidean) norm of the difference vectors. Hence the selection of the best matching unit is dependent on every input fed to the map starting from the beginning of the series. This suggests that the updating rule for the parameters of the map should include the whole history of the inputs. In this case, however, it is noticed that the effect of the feedback goes quickly towards zero in time. Since we use limited accuracy in calculations, this occurs with certain number of iterations. Hence the difference vector can be approximated by using certain number of past samples of the input vectors.

Based on this approximation, the learning algorithm of RSOM is implemented as follows. The map is presented an *episode* of consecutive input vectors starting from a random point in the data. The number of vectors belonging to the episode is dependent on the leaking coefficient α used in the units. At the end of the episode, the impulse response $h(k)$ of the recurrent filter (see Fig. 1) is below 5 % of the initial value. The best matching unit is selected at the end of the episode based on the norm of the difference vector. The updating of the vector and its neighbors is carried out as in Eq. 2. After the updating, all difference vectors are set to zero, and a new random starting point from the series is selected. The above scenario is repeated until the mapping has formed.

Figure 2. shows the procedure for building the models and evaluating their prediction abilities with testing data. The time series is first divided to training and testing data. Input vectors to

RSOM are formed by windowing the time series. For model selection purposes the training data is divided to cross-validation data sets. Here we will use v-fold cross-validation [9], in which the training data is first divided to v disjoint subsets. One subset at a time is put aside, and the model is trained using the union of the remaining v-1 subsets, then tested for the subset left out. This procedure is carried out for each subset, and cross-validation error is gained as the mean of the errors of the subsets. Cross-validation error is used as an estimate of the true prediction error of the model.

In training phase the free parameters of RSOM include length of the input vectors, time step between input vectors, leaking coefficient in the units and the number of RSOM units. The episode length is dependent on the leaking coefficient as described earlier. After the training RSOM is used to divide the data into local data sets according to the best matching unit on the map. Linear regression models, that have the same number of parameters as the vectors of RSOM units, are then estimated using these local data sets. The prediction error for each model consisting of RSOM and associated local models is estimated using cross-validation error. Finally, the best model according to cross-validation error is selected. The selected model is trained again with the whole training data. This model is then used to predict the test data set that has not been presented to the model before.

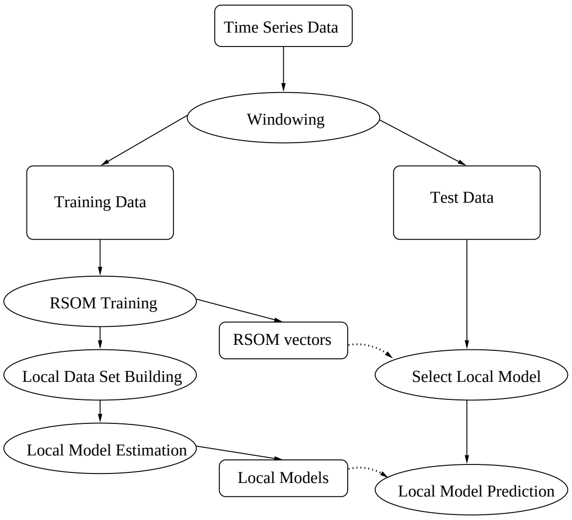

Fig. 2. Construction of the local models.

Next we will make a comparative case study to determine how the RSOM approach as described performs in time series prediction tasks.

III. Case Studies

Three different time series were studied to compare RSOM with local linear models to MLP and AR models. The prediction task was in all cases one-step prediction. From each series four cross-validation data sets were generated. The same data sets were used for all models to prevent biasing the results based on different data sets.

The RSOM with local linear models were estimated as follows: The input vector length p, time step between consecutive input vectors s, number of units n_u and the leaking coefficient α of the units in the map were free parameters giving rise to model $RSOM(p, s, n_u, \alpha)$. In the studied cases the number of RSOM units was varied as $n_u \in \{5, 9, 13\}$ and the leaking coefficient was varied as $\alpha \in \{1.0, 0.95, 0.78, 0.73, 0.625, 0.45, 0.40, 0.35\}$ corresponding to episode lengths 1 ... 8. The time step between consecutive vectors was varied as $s \in \{1, 3, 5\}$ while the length of the input vector p was varied differently in the three cases as described later. The number of regression models used equals the number of RSOM units n_u. The order of the regression models equals the length of the vectors p in RSOM units. The regression models were estimated using least squares optimization algorithm implemented with MATLAB 5 statistics toolbox. Local regression models associated with each unit of RSOM were estimated using the data for which the corresponding RSOM unit was the best matching unit.

The MLP network was trained with Levenberg-Marquardt learning algorithm implemented with MATLAB 5 neural network toolbox. An *MLP(p,s,q)* network with one hidden layer, p inputs and q hidden units was used. In this s is the time step between consecutive input vectors. In order to make comparison of the results fair, parameters p and s were chosen to be the same as in RSOM models. The number of hidden units was varied as $q \in \{3, 5, 7, 9\}$. The training was performed five times for each *MLP(p,s,q)* model and for each cross-validation data set with different initial weights. The smallest errors of each fold of the cross-validation data sets were used to get an estimate of the prediction error. The model that gave the smallest cross-validation error was selected for evaluation with the test data.

AR(p) models with p inputs were estimated with MATLAB 5 using the least-squares algorithm. The order of the AR model was varied as $p \in \{1, \ldots, 50\}$. The same cross-validation scheme as with other models was used. In this case the prediction errors serve as an example of the accuracy of a global linear model in the current tasks.

A. Mackey-Glass Chaotic Series

The first of our test cases is the Mackey-Glass time series, produced by a time-delay difference system of the form [17]:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\gamma)}{1 + x(t-\gamma)^{10}} \qquad (15)$$

where $x(t)$ is the value of the time series at time t. This system is chaotic for $\gamma > 16.8$. In the present test the time series was constructed with parameter values $\alpha = 0.2$, $\beta = -0.1$ and $\gamma = 17$ and it was scaled between [-1,1]. From the beginning of the series shown in Fig. 3. 3000 samples was selected for training, and the rest 1000 samples were used for testing. For RSOM and MLP models length of the the input vector was varied as $p \in \{3, 5, 7\}$.

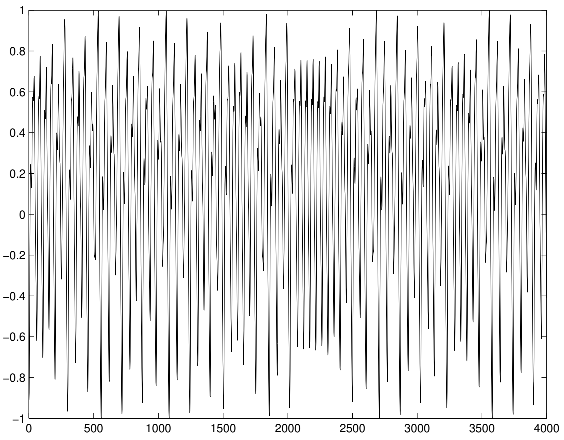

Fig. 3. Mackey-Glass chaotic time series.

The sum-squared errors gained for one-step prediction task are shown in Table I. There it is seen that the *MLP*(3,1,7) model gives the smallest cross-validation error but fails to predict the test set accurately. For the *AR*(2) model the results are opposite. AR model does not model here the underlying phenomena, instead it predicts the next value of the series using mainly the previous value. This scheme gives in this case the best prediction accuracy. *RSOM*(3,1,5,0.95) gives moderate accuracy for both cross-validation and test data sets. With the test set, however, the error is smaller than with MLP network.

B. Laser Series

The second of our test cases is the laser time series which consists of measurements of the intensity of an infrared laser in a chaotic state. Explicit measurement conditions and the theory of the chaotic behavior in the series is given in [29]. The data is available from an anonymous ftp server [1].

TABLE I
ONE-STEP PREDICTION ERRORS FOR MACKEY-GLASS TIME SERIES.

	CV Error	Test Error
RSOM(3,1,5,0.95)	6.5556	3.1115
MLP(3,1,7)	0.3157	3.7186
AR(2)	4.1424	1.600

The training data contains first 2000 samples of the laser series as depicted in Fig. 4. For the testing the rest 1000 samples were used. Both series were scaled between [-1,1]. For RSOM and MLP models length of the the input vector was varied as $p \in \{3, 5, 7\}$.

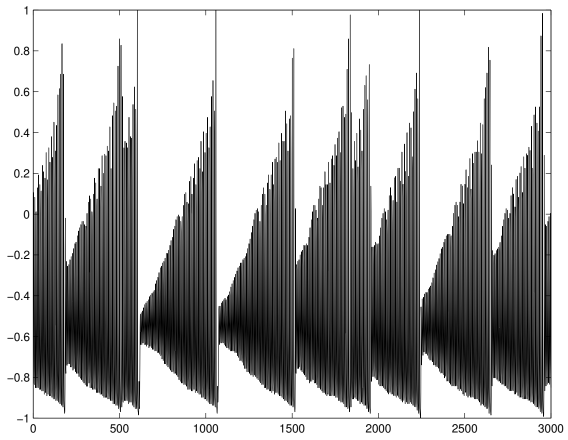

Fig. 4. Laser time series.

The sum-squared errors gained for one-step prediction task are shown in Table II. The laser series is highly nonlinear and thus the errors gained with *AR*(12) model are considerably higher than for other models. The series is also stationary and almost noiseless, which explains the accuracy of the *MLP*(9,1,7) model predictions. In this case *RSOM*(3,3,13,0.73) gives results that are better than with AR model but worse than with MLP model.

TABLE II
ONE-STEP PREDICTION ERRORS FOR LASER TIME SERIES.

	CV Error	Test Error
RSOM(3,3,13,0.73)	14.6995	7.3894
MLP(9,1,7)	4.9574	0.9997
AR(12)	69.8093	29.8226

[1] ftp://ftp.cs.colorado.edu/pub/Time-Series/SantaFe/ containing files A.dat (first 1000 samples) and A.cont (as a continuation to A.dat 10000 samples)

C. Electricity Consumption Series

The third of our cases in an electricity consumption series which contains measured load of an electric network. Measurements contain hourly consumption of electricity over a period of 83 days (2000 samples). The series was scaled between [-1,1], see Fig. 5. For the training 1600 samples were selected, and the rest 400 samples were used for testing. For RSOM and MLP models length of the the input vector was varied as $p \in \{4, 8, 12\}$.

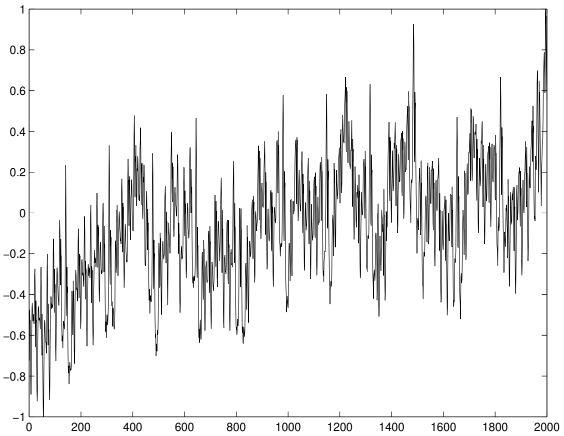

Fig. 5. Electricity consumption time series.

The sum-squared errors gained for one-step prediction task are shown in Table III. The electricity consumption series of Fig. 5. contains 24 hours long cycle and also slower trend and noise in the form of measurement errors. $AR(30)$ model is found to reach quite acceptable results, due to the fact that model includes the whole 24 hour cycle. As the results with $MLP(8,1,9)$ model show, nonlinear model can reach better predictions with a shorter window length. In this case $RSOM(8,1,13,0.73)$ model does not give any improvement due to the insufficient input vector length used in model estimation.

TABLE III
One-step Prediction Errors for Electricity Consumption Time Series.

	CV Error	Test Error
RSOM(8,1,13,0.73)	18.0673	2.6735
MLP(8,1,9)	7.6007	1.4345
AR(30)	6.5698	2.1059

IV. Conclusions

Time series prediction using Recurrent SOM with linear regression models has been presented. For the selected prediction tasks RSOM with local linear models gives promising results. In the cases studied here it models the series better than linear models. In the case of the Mackey-Glass series, the AR results cannot be considered better than the results of RSOM since AR does not really model the series. Because we used local linear models in RSOM, it can be argued that its prediction accuracy should be at least the same as with pure linear models. Due to the selection of RSOM parameters, this was not achieved in all cases. However, in the case of the highly nonlinear laser series RSOM model gave considerably better prediction results than linear models. In the studied cases MLP seems to perform better than RSOM. This is mainly due to the selected one-step prediction problem, where MLP could yield the underlying mapping effectively. Another reason for RSOM not achieving as good results as MLP is the linear models used.

What makes the RSOM model attractive to the study of time series is the built-in property of the SOM to construct topological maps from the data. In this study we used one-dimensional RSOM and did not use the visualization possibilities of the map. In the case of two-dimensional map, the state changes of the process can be visualized with the location of the best matching unit in the map as a function of time. It could be expected that this visualization capability is perhaps the most important point in using the RSOM instead of other clustering algorithms.

Another attractive property of RSOM is the unsupervised learning of the context from the data. It allows building models from the data with only a little *a priori* knowledge. This property also allows using RSOM as an explorative tool to find statistical temporal dependencies from the process under consideration.

In this study we used RSOM that has the same feedback structure in all the units. It is possible, however, to allow the units of RSOM to have different recurrent structures. This kind of structure could yield topological maps of different temporal contexts in the series. Such extensions of RSOM will be researched in the near future.

Acknowledgments

This study has been funded by *The Academy of Finland.*

References

[1] P. Bak. *How Nature Works.* Springer Verlag, New York, 1996.

[2] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control.* Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[3] G.J. Chappell and J.G. Taylor. The temporal Kohonen map. *Neural Networks*, 6:441–445, 1993.

[4] M. Cottrell. Theoretical aspects of the SOM algorithm. In *Proc. of Workshop on Self-Organizing Maps*, pages 246–267. Helsinki University of Technology, 1997.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[6] J. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–142, 1991. With discussion.

[7] N. Gershenfeld and A. Weigend. The future of time series: Learning and understanding. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 1–70. Addison-Wesley, 1993.

[8] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, New York, 1994.

[9] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja. Neural and statistical classifiers—taxonomy and two case studies. *IEEE Trans. Neural Networks*, 8(1):5–17, 1997.

[10] M. Jordan and A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[11] R. Kalman. A new approach to linear filtering and prediction problems. *ASME Transactions*, 82D, 1960.

[12] J. Kangas. *On the Analysis of Pattern Sequences by Self-Organizing Maps.* PhD thesis, Helsinki University of Technology, May 1994.

[13] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, Heidelberg, 1989.

[14] J. Lampinen and E. Oja. Self-organizing maps for spatial and temporal AR models. In *Proc. of SCIA'89, 6th Scandinavian Conf. on Image Analysis*, 1989.

[15] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and signal modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.

[16] M. Lehtokangas. *Modeling with Layered Feedforward Neural Networks.* PhD thesis, Tampere University of Technology, Tampere, Finland, 1995.

[17] M. Mackey and L. Glass. Oscillations and chaos in physiological control systems. *Science*, pages 197–287, 1977.

[18] T. Martinetz, S. Berkovich, and K. Schulten. 'Neural-Gas' network for vector quantization and its application to time-series prediction. *IEEE Trans. Neural Networks*, 4(4):558–569, 1993.

[19] M. Mozer. Neural net architectures for temporal sequence processing. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 243–264. Addison-Wesley, 1993.

[20] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 2nd edition, 1984.

[21] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications.* MacMillan Publishing Company, 1992.

[22] H. Ritter and K. Schulten. Convergence properties of Kohonen's topology preserving maps: Fluctuations, stability and dimension selection. *Biological Cybernetics*, 60:59–71, 1988.

[23] A. Singer, G. Wornell, and A. Oppenheim. A nonlinear signal modeling paradigm. In *Proc. of the ICASSP'92*, 1992.

[24] H. Tong. *Nonlinear Time Series: A Dynamical System Approach.* Oxford University Press, 1990.

[25] A. Tsoi and A. Back. Locally recurrent globally feedforward networks: A critical review of architectures. *IEEE Transactions on Neural Networks*, 5(2):229–239, 1994.

[26] M. Varsta, J. Heikkonen, and J. Del Ruiz Millán. Context learning with the self-organizing map. In *Proc. of Workshop on Self-Organizing Maps*, pages 197–202. Helsinki University of Technology, 1997.

[27] J. Vesanto. Using the SOM and local models in time-series prediction. In *Proc. of Workshop on Self-Organizing Maps*, pages 209–214. Helsinki University of Technology, 1997.

[28] J. Walter, H. Ritter, and K. Schulten. Non-linear prediction with Self-Organizing Maps. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 1, pages 589–594, 1990.

[29] A. Weigend and N. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993.

[30] G. Yule. On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Philos. Transactions of the Royal Society*, A226:267–298, 1927.