Inductive Modelling of Temporal Sequences by Means of Self-organization

Jan Koutnik[1]

[1]*Computational Intelligence Group,*
Dept. of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo nám. 13, 121 35 Praha 2, Czech Republic

koutnij@fel.cvut.cz

Abstract. *In this paper we present a new self-organizing neural network, which builds a spatio-temporal model of an input temporal sequence inductively. The network is an extension of Kohonen's Self-organizing Map with a modified Hebb's rule for update of temporal synapses. The model building behavior is shown on inductive learning of a transition matrix from a data generated by a Markov Chain.*

Keywords

Inductive modelling, self-organization, temporal sequences.

1 Introduction

In processing of temporal sequences, there are basically four tasks to solve. First, a model of the sequence is created. I the second task, the model is exploited for recognition of temporal sequences. Next task is a prediction of forthcoming data from the past sequence using the model. In the last task the model is used for generation of data that are more or less like the sequence on which the model was trained.

In good old fashioned artificial intelligence Hidden Markov Models (HMM) [7] are widely used for modelling, recognition, prediction and generation of temporal sequences with a great success.

In computational intelligence artificial neural networks that are capable of effective processing of temporal sequences started to appear in 80's. A key structure in the network architectures is a recurrent connection that allows effective dealing with a temporal sequence. Neural networks can be divided into two categories – supervised neural networks and unsupervised neural networks. The supervised neural networks can be used for recognition of temporal sequences. Unsupervised neural networks can be exploited in temporal sequences modelling, preprocessing and prediction.

There are gradient algorithms that can train weights of a recurrent layered structure like Back Propagation Through Time [14]. The training algorithm trains a network unrolled back in time. Authors of [3] show that the temporal memory of the network rapidly vanishes in time. There are Elman and Jordan networks [1], which are models of sequential automata trained by a gradient algorithm.

Prediction performance of the networks and the training algorithms is usually compared on regular and context-free and context-sensitive language prediction. Recent variant of Long-short Term Memory (LSTM) [2] can bridge long time lags in a regular grammar ($a^n b^n$) strings.

In the field of unsupervised neural networks, there are several models that raised from basic Kohonen's Self-organizing Map (SOM) [4]. Two basic models named Temporal Kohonen Map (TKM) [10] and Recurrent Self-organizing Map (RSOM) [9] are similar models with a recurrent self-connection for each neuron in the network layer. There are several recent networks that employ Hebbian synapses between basic Kohonen's layer and another layer [6].

We present a simple extension of basic Kohonen's SOM with a layer of synapses (connections) trained by a Hebb's rule. We added no layer to the network. It is the most simple extension of the network possible. The network was named Temporal Hebbian Self-organizing Map (THSOM).

The paper is organized as follows. Second section presents related self-organizing recurrent neural networks. Third section introduces the THSOM neural network as a model for temporal sequences. Fourth section shows a simulation of how the network build the model inductively from the data. Fifth section discusses the results and sixth section concludes the paper.

2 Related Work

There are several variants of processing of previous state of the network.

Temporal Kohonen Map and Recurrent SOM exploit a context stored in an activation of the neuron in previous time step by a recurrent self-connection. A study of the behavior of the neural network is not easy and extraction of temporal model from the network is not straightforward [11].

Recursive SOM (RecSOM) [12] saves all neuron activations for further time steps in a vector handled by Euclidean metric as well as spatial weights. The disadvantage of the RecSOM is the dimensionality of the input vector enriched by the context from previous time step.

SOMSD [13] uses only a last winner neuron index as a representative of the context.

The most powerful variant of recurrent SOM is the Merging SOM (MSOM) [8], which contains two weight vectors, one for spatial input and second for the context. The context in MSOM is calculated weight vector of previous winner and it's context.

3 Temporal Hebbian Self-organizing Map (THSOM)

We introduce Temporal Hebbian Self-organizing Map (THSOM) as a most simple SOM variant, which contains two maps – spatial map and temporal map. The temporal map is trained according to modified Hebbian rule. The spatial map uses standard Kohonen's training algorithm. THSOM exploits whole context similarly to RecSOM but the context data are not measured by Euclidean metric and therefore suffer from dimensionality problem.

The THSOM architecture consists of one layer (grid) of neurons. The neurons are fully connected to input vector of dimension d, these connection make up the spatial map. The neurons are connected to all neurons in the grid using recurrent temporal synapses (temporal map). The neurons are of a hybrid type. They contain two types of similarity measures, Euclidean metrics for measuring similarities in input spatial space and scalar product for measuring similarities in temporal space. THSOM architecture is depicted in 1

3.1 THSOM Neuron

The THSOM layer consists of hybrid neurons. The neurons are considered hybrid because usage of two similarity measures. Input vector of the networks is measured using Euclidean distance, temporal map is processed using scalar product of the temporal weight vector and vector of activations of neurons in previous time step, see following equation:

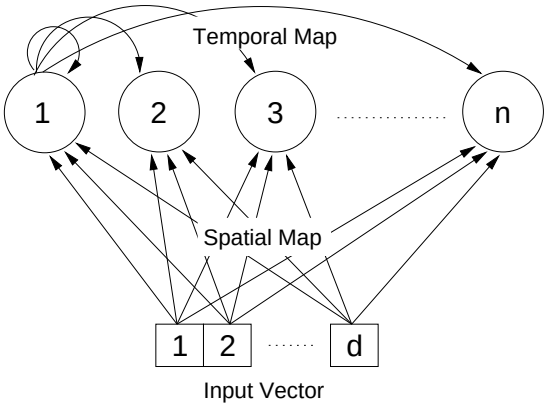

Fig. 1. THSOM Architecture. Neurons are depicted as circles. There is n neuron in one network layer. The input vector of dimension d is lead to the network via spatial map synapses. The spatial map makes full connection between the input vector and neurons. The temporal map makes a full connection among all neurons. There are depicted connections for neuron 1 in the schematic drawing.

$$y_i^t = \sqrt{d} - \sqrt{\sum_{j=1}^{d}(x_j^t - w_{ij}^t)^2} + \sum_{k=1}^{n}(y_k^{t-1} m_{ik}^t) \tag{1}$$

where y_i^t is activation (output) of i-th neuron in time t, $\sqrt{d}$ is a constant square root of input vector dimension d, x_j^t is j-th networks input in time t, w_{ij}^t is spatial weights for j-th input in time t, y_k^{t-1} is output of k-th neuron in time $t-1$, m_{ik}^t is temporal weights for k-th neuron in time t (from neuron k to neuron i), n is a number of neurons in network, and d is a dimension of input space (vector).

All neuron activations are normalized after the time step using equation:

$$y_i^t = \frac{y_i^t}{max(y_i^t)} \tag{2}$$

3.2 Training of the Spatial and Temporal Maps

The spatial weights are updated using a rule used in classical SOM (Kohonen's Self-organizing Map). The weights are updated in order to move the neuron in the space closer to the input vector. First, the winning neuron called Best Matching Unit (b) index is calculated:

$$b = \arg\max_i(y_i^t) \tag{3}$$

Afterwards, all neuron spatial weights are updated using following rule:

$$w_{ij}^{t+1} = w_{ij}^t + \alpha\eta_{ib}(x_j^t - w_j^t) \tag{4}$$

where α is a learning rate (multiplication constant) end η_{ib} is a neighborhood distance function between currently updated i and best matching unit b neurons induced by the network grid.

As a function η_{ib} we have chosen *cutgauss* neighborhood function calculated using equation:

$$\eta_{ib} = \begin{cases} \alpha e^{\frac{-\gamma^2}{2\sigma^2}} & \gamma^* \leq \nu \\ 0 & \gamma^* > \nu \end{cases} \tag{5}$$

where γ is an Euclidean distance of neurons measured in the neuron layer grid, ν is neighborhood size, which diminishes during the learning, α is the learning rate, which diminishes during the learning and σ is a steepness of the Gaussian neighborhood function.

Parameter γ is calculated according to topology of the neuron grid using following equation:

$$\gamma = \sqrt{\sum_{g=1}^{G} (i_g - b_g)^2} \tag{6}$$

where G is a dimension of the neuron grid (usually a grid with $G = 2$ is used) and i and b are the neurons for which the distance is calculated.

Parameter γ^* is a maximum distance in the grid in a particular dimension when a neighborhood of rectangular shape is used:

$$\gamma^* = \max_g |i_g - b_g| \tag{7}$$

There are several shapes of the neighborhood cut and the neighborhood function itself. We used neighborhood cut of rectangular shape induced by 7 and rounded Gaussian shape described by 6. The neighborhood diameter should be diminished during the training.

Temporal weights use modified Hebbian learning rule according to following formulas:

$$m_{ik}^{t+1} = \begin{cases} \min(\max(m_{ik}^t + \alpha(1 - m_{ik}^t + \beta), K_l), K_h) & \text{for } k = \arg\max_k(y_k^{t-1}) \\ \min(\max(m_{ik}^t - \alpha(m_{ik}^t + \beta), K_l), K_h) & \text{otherwise} \end{cases} \tag{8}$$

where y_k^{t-1} activation of a neuron after previous time step, β and α control temporal synapses learning rate, α can start on 1.0 and is decreased in time for slow down oscillation around desired temporal weight value, K_l is a low boundary of temporal weights, usually ($K_l = 0$), K_h is a high boundary of temporal weights, usually ($K_h = 1$).

The input vectors should be in $< 0, 1 >$ interval for proper function of the hybrid neuron.

4 Experimental Results

In experimental results we used a probabilistic automaton (Markov Chain, MC) to generate sequences that were presented to the network input, each vector from the sequence in one time step. First experiment demonstrate a capability of THSOM to induce spatial and temporal map from the data in a simple experiment. The MC generates a symbol in each state. The symbols are vectors in two-dimensional space and are generated according to matrix $\mathbf{S}$. Matrix $\mathbf{A}$ is a transition probability matrix, where each row contains probabilities of transition to state in corresponding column. Vector $\mathbf{I}$ contains initial probabilities of states.

In a simple simulation we have used a 2-dimensional TSOM grid consisting of 4 (2x2) neurons. The learning is split into two parts, in each part the input signal ($\mathbf{S}$) was repeated to the network 5 times. The neighborhood size (ν) was diminished from 1 (whole network) to 0 (BMU only). All weights were randomized before the experiment.

The input signal is a temporal sequence of four two-dimensional vectors:

$$\mathbf{S} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \mathbf{A} = \begin{pmatrix} 0.2 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.8 \\ 0.8 & 0.0 & 0.0 & 0.2 \end{pmatrix} \mathbf{I} = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

We trained the network in two phases. In the first phase, we performed 100 steps for neighborhood covering the whole network. In the second phase, the neighborhood radius was decreased to 0 (only the winner neuron weights are modified).

In each step a symbol represented by a row vector from $\mathbf{S}$ is added a Gaussian noise with $\sigma = 0.02$.

After learning, following spatial ($\mathbf{W}$) and temporal ($\mathbf{M}$) matrices were obtained:

$$\mathbf{W} = \begin{pmatrix} 0.99 & 0.0 \\ 0.0 & 1.0 \\ 1.0 & 0.99 \\ 0.01 & 0.0 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 0.18 & 0 & 0.82 & 0 \\ 0 & 0.18 & 0 & 0.82 \\ 0 & 0.75 & 0.25 & 0 \\ 0.74 & 0 & 0 & 0.26 \end{pmatrix}$$

As we can see, the spatial map contains neurons put to same positions as positions of input vectors in $\mathbf{S}$ in the sequence. Spatial map $\mathbf{M}$ contains values close to 0.2 on the main diagonal. Values close to 0.8 are placed on column for a neuron – state from which a transition to corresponding neuron expressed by a row is the most probable.

Neuron no.3 is mapped to position (according to spatial weight) 00, the most probable transition (expressed by a highest temporal weight) leads to neuron no.2 using temporal weight 0.2 (second row, last column) etc.

The temporal weight matrix can be transformed to transition probability matrix of a Markov Chain using transposition and normalization. The transposed matrix is normalized linearly to have sum of probabilities in a row equal to 1:

$$\mathbf{T_2} = \begin{pmatrix} 0.2 & 0 & 0 & 0.8 \\ 0 & 0.19 & 0.81 & 0 \\ 0.77 & 0 & 0.23 & 0 \\ 0 & 0.76 & 0 & 0.24 \end{pmatrix}$$

The only difference to MC that generated the sequence is in different mapping of neuron numbers to input vectors. We can see, that neuron no. 0 represents input vector no. 3, neuron no. 1 represents input vector no. 1, neuron no. 2 represents vector no. 2 and neuron no. 3. represents vector no. 0. An arbitration of neurons to the input vectors could not be influenced. The matrix of resulting MC can be rearranged to proper positions as well.

Figure 2 displays induction of the model in the learning phase. There is an unorganized model after random initialization. We can see how the neurons are moved to proper positions and the temporal weights are strengthened between neurons that are sequentially activated.

Building of spatio-temporal model is depicted in figure 2. The input is two dimensional, therefore the graph is placed into a two dimensional plane. THSOM neurons are depicted as circles. The spatial map weights are represented by solid arrows. Arrow strength represents a temporal weight value. Dark thick arrow is a K_h weight and nearly invisible arrow represents K_l weight. Each neuron's inner circle color represents self-weight (m_{ii}), black circle represents K_h weight. Neuron positions represent spatial weights in two-dimensional input space. Crosses represent input vectors. Consequences between input

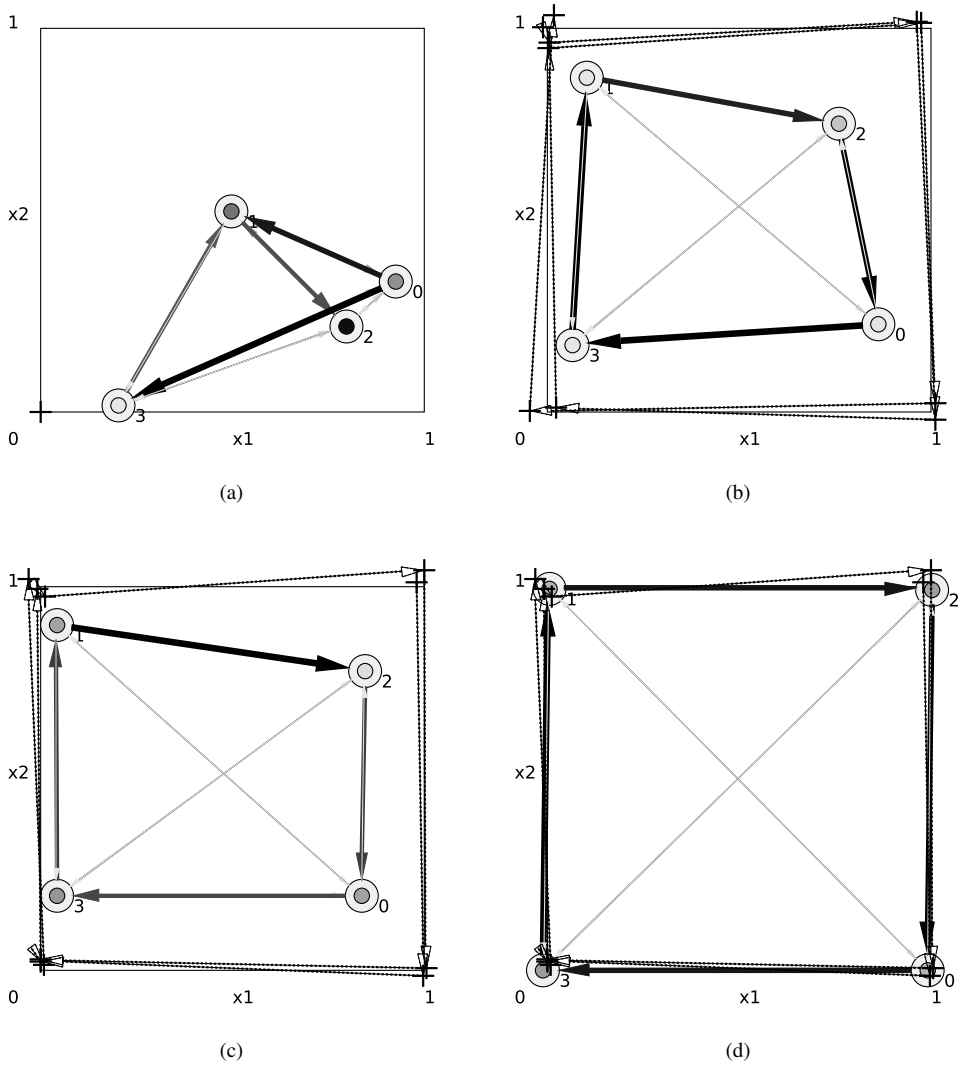

Fig. 2. Figures display building of spatio-temporal model. THSOM neurons are depicted as circles. The spatial map weights are represented by solid arrows. Arrow strength represents a temporal weight value. Each neuron's inner circle color represents self-weight Crosses represent input vectors. Consequences between input vectors are represented by dashed arrows. In each sub-figure, last 10 input vectors are included. Sub-figures display spatial and temporal maps after 0 – initialized (a), 10 (b), 100 (c) and 200 (d) steps.

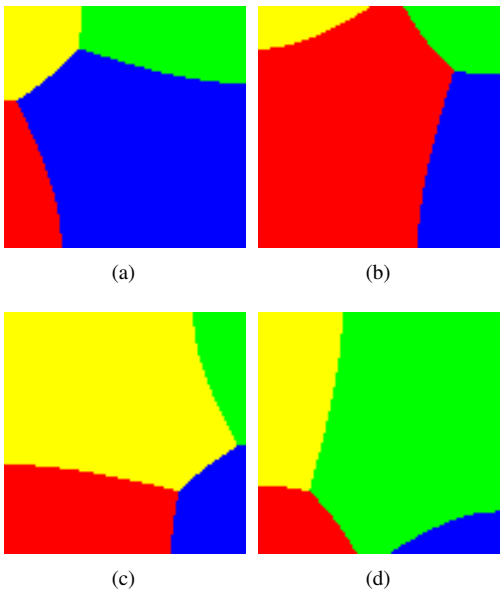

Fig. 3. State affected Voronoi plots display how a BMU for further vectors from the sequence will be selected. There are four planes represented by a different color in each Voronoi plot. There are four Voronoi plots for four network states. We can see how the planes change their shape according to the network state. For example in sub-figure (a) neuron no. 2 is the BMU. Due a strong temporal synapse leading to neuron no. 0 a Voronoi plane for that neuron is the biggest one.

vectors are represented by dashed arrows. In each sub-figure, last 10 input vectors are included. Sub-figure 2(a) displays randomly initialized model with random temporal and spatial weights. Sub-figure 2(b) displays the model after 10 steps. Sub-figure 2(c) displays the model after the end of the first rough phase. The model does not fit the data well because of neighborhood function in the spatial weights training. The same behavior can be seen in original SOM. Last sub-figure 2(d) displays THSOM after 200 steps at the end of training phase. We can see that the neurons are placed into correct positions and the temporal model is induced properly.

There are *State affected Voronoi plots* in Figure 3. Normally, the four Voronoi sub-planes will be of a square shape and same dimension because the neurons are placed in the edges of the two dimensional input space hypercube. Since the temporal map influent further BMUs selected for further vectors taken from the input sequence, the Voronoi plots are modified depending on a current state of THSOM.

Sub-figure 3(a) contains a Voronoi plot for BMU neuron no. 2 in time t. Due a strong temporal synapse expressed by weight $m_{0,2}$ in matrix $\mathbf{M}$ or transition probability in gained Markov Chain ($\mathbf{T}_{\mathbf{2}_{2,0}}$), the Voronoi sub-plane for neuron no. 0 is bigger that the other sub-planes. The Voronoi plot shape changes as the network changes it's state.

5 Discussion

THSOM can act as a spatial model for the data. It works as a normal Kohonen's self-organizing map. When temporal weights are non-zero, a state of the network is considered in BMU search in further time step.

5.1 THSOM as a signal model

A signal model of possibly noisy data can be obtained inductively using THSOM. The temporal and spatial maps are split into two separate weight matrices. Advantage is that a transition matrix of induced Markov Chain can be easily obtained by transposition and normalization of temporal weight matrix. THSOM can contain either less neurons than expected states of the temporal sequence. Therefore it can be used for compression of the sequence by representation of more than one similar input vectors in a single neuron as a standard SOM. THSOM can be used for reduction of dimensions of the input vector space as well as standard SOM. The model behavior in training respects self-organizing map property of preservation the input data shape topology. In a simple experiment in this paper the number of neurons is the same as number of states in the MC used for generation of the input data. In second phase of the training a neighborhood a neuron is reduced to the neuron itself and the neurons move to centers of input vector clusters. Finally, the spatial map contains spatial description of vector in the input sequence and the temporal map contains temporal description of the sequential order of the input vectors in the sequence.

5.2 Signal Prediction

Signal prediction with THSOM can be easily realized. In each state the obtained MC predicts further state as a biggest value in the current state row of matrix $\mathbf{T_2}$. The most probable predicted input vector is then decoded using spatial weight matrix for the corresponding neuron index. The most probable attractor can be predicted from the current state by not taking spatial connections into account.

Signal recognition task is not possible by THSOM itself. THSOM is not capable to assign a neuron to a sequence. One can use TICALM [5] model for recognition which recognizes particular sequence by accepting it with high speed convergence for trained sequences in competition realized by laterally inhibited neurons layer.

6 Conclusion

We designed a single layer neural network based on Self-organizing map with hybrid neurons. The network was named THSOM (Temporal Hebbian SOM). THSOM measures similarities in spatial input using Euclidean metric and similarities in time component of the signal with scalar product associative synapses. The network is suitable for signal modelling, and prediction. Temporal model of the sequence can be directly extracted from the spatial weight matrix trained by Hebb's rule. Performance of the model was tested on inductive reconstruction of Markov Chain from a sequence of vectors generated by another Markov Chain.

THSOM exploits temporal context as a complete network state in previous time step. The context is processed in hybrid neurons with scalar product and therefore suffer from a dimensionality problem.

Acknowledgements

This research is supported by the research program "Transdisciplinary Research in the Area of Biomedical Engineering II" (MSM6840770012) sponsored by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[2] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[3] Sepp Hochreiter and Jürgen Schmidhuber. LSTM can solve hard long time lag problems. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 473. The MIT Press, 1997.

[4] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, 3rd edition, 2001.

[5] J. Koutnik and M. Snorek. Self-organizing neural networks for signal recognition. In *16th International Conference on Artificial Neural Networks Proceedings (ICANN 2006), Part I*, volume 4131/2006, pages 406–414. Springer Berlin / Heidelberg, 2006.

[6] A. Nyamapfene and K. Ahmad. Unsupervised multi-net simulation: An application to child development. In *Proceedings of the IJCNN '06. International Joint Conference on Neural Networks*, pages 1427–1433, 2006.

[7] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77(2):257– 286*, 77(2):257–286, 1989.

[8] M. Strickert and B. Hammer. Neural gas for sequences. In T. Yamakawa, editor, *Proceedings of the Workshop on Self-Organizing Networks (WSOM)*, pages 53–58. Kyushu Institute of Technology, 2003.

[9] J. Heikkonen K. Kaski T. Koskela, M. Varsta. Temporal sequence processing using recurrent SOM. In *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Engineering Systems*, volume 1, Adelaide, Australia, 1998.

[10] M. Varsta, J. Heikkonen, and J. Lampinen. Analytical comparison of the temporal kohonen map and the recurrent self organizing map. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 273–280. D-facto Publications, 2000.

[11] Markus Varsta. Self-organizing maps in sequence processing, 2002.

[12] T. Voegtlin. Context quantization and contextual self-organizing maps. In *Proc. Int. Joint Conf. on Neural Networks*, volume 5, pages 20–25, 2000.

[13] Thomas Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15(8-9):979–991, 2002.

[14] P. J. Werbos. Back propagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560, 1990.