Self-Organizing Maps for Time Series

Barbara Hammer[1], Alessio Micheli[2], Nicolas Neubauer[3], Alessandro Sperduti[4], Marc Strickert[5]

[1]Clausthal University of Technology, Computer Science Institute, Clausthal-Zellerfeld, Germany, hammer@in.tu-clausthal.de
[2]Università di Pisa, Dipartimento di Informatica, Pisa, Italy
[3]University of Osnabrück, Institute of Cognitive Science, Osnabrück, Germany
[4]Università di Padova, Dipartimento di Matematica Pura ed Applicata, Padova, Italy
[5]Institute of Plant Genetics and Crop Plant Research Gatersleben, Pattern Recognition Group, Gatersleben, Germany

Abstract - *We review a recent extension of the self-organizing map (SOM) for temporal structures with a simple recurrent dynamics leading to sparse representations, which allows an efficient training and a combination with arbitrary lattice structures. We discuss its practical applicability and its theoretical properties. Afterwards, we put the approach into a general framework of recurrent unsupervised models. This generic formulation also covers a variety of well-known alternative approaches including the temporal Kohonen map, the recursive SOM, and SOM for structured data. Based on this formulation, mathematical properties of the models are investigated. Interestingly, the dynamic can be generalized from sequences to more general tree structures thus opening the way to unsupervised processing of general data structures.*

Key words - Self-organizing maps, time series, merge SOM, recurrence, fractal, encoding, structures

1 Introduction

Biological information processing systems possess remarkable capacities with respect to accuracy, speed, noise tolerance, adaptivity and generalization ability for new stimuli, which outperform the capability of artificial systems. This ability is astonishing as the processing speed of biological nerve cells is orders of magnitudes slower than the processing speed of transistors in modern computers. To reach this power, biological systems rely on distributed and parallel processing and highly efficient representation of relevant stimuli within the cells. Self-organization plays a major role to reach this goal. As demonstrated in numerous simulations and applications [13, 17], self-organizing principles allow the development of faithful topographic representations leading to clusters of given data, based on which an extraction of relevant information and supervised or unsupervised postprocessing is easily possible.
Stimuli typically occuring in nature have a time characteristic: data from robotics, sensor streams, speech, EEG and MEG, or other biological time series, to name just a few. However, most classical unsupervised models are restricted to vectorial data. Thus, the question arises, how signals with a specific time characteristic can be faithfully learned using the powerful principles of self-organization. Several extensions of classical self-organizing models exist for

dealing with sequential data, involving, for example

1. fixed length time windows as used e.g. in [19, 24];
2. specific sequence metrics, e.g. operators or the edit distance [5, 17, 18, 25]; thereby, adaptation might be batch or online;
3. statistical modeling incorporating appropriate generative models for sequences such as proposed in [2, 31];
4. mapping of temporal dependencies to spatial correlation, e.g. as traveling wave signals or potentially trained temporally activated lateral interactions [4, 23, 34];
5. recurrent processing of the time signals and recurrent winner computation based on the current signal and previous activation [3, 6, 11, 12, 15, 28, 29, 32, 33]

Many of these approaches have been proposed recently, demonstrating the increasing interest in unsupervised learning models for time series. A more detailed overview is provided e.g. in [1]. However, there does not yet exist a 'canonical' model or notation which captures the main aspects of unsupervised sequence processing in a common dynamic. In addition, the capacity of these models and their mutual relation is hardly understood. Thus, there is a need for a unification of the notation and an exact mathematical investigation and characterization of the benefits and drawbacks of these models.
Here, we will focus on *recurrent* self-organizing models. For recurrent models, a unifying notation can be found [10] which allows the formalization of the important aspects within one unified dynamical equation and which identifies a crucial part of model design: recurrent models essentially differ in the *context*, i.e. the way how sequences are internally represented. This internal representation of sequences severely influences the processing speed, the flexibility with respect to the neuron topology, and the capacity of the models. Interestingly, these issues can be investigated in an exact mathematical way in many cases [8, 9]. We will discuss this fact for a recent recurrent model, the Merge SOM (MSOM) [28], in detail. Afterwards, we review (some) known results about the capacity of recurrent models, and we conclude with a short look at a generalization of this approach for more general data structures.

2 MSOM

The MSOM has been proposed in [28] as an efficient and flexible model for unsupervised processing of time series. The goal of unsupervised learning is to represent a set of stimuli faithfully by a set of neurons (prototypes). We are interested in stimuli with a temporal characteristic, i.e. one or more time series of the form $(\mathbf{s}^1, \ldots, \mathbf{s}^t, \ldots)$ with elements $\mathbf{s}^t \in \mathbb{R}^n$. Time series are processed recursively, i.e. the entries $\mathbf{s}^t$ are fed consecutively to the map, starting from $\mathbf{s}^1$. A neural map is given by a set of neurons or prototypes $\{1, \ldots, N\}$ The neurons are characterized by a neighborhood structure (often a regular two-dimensional lattice) which can be used for visualization and neighborhood cooperation during training, and a weight which represents a typical stimulus. Given an input, the weight vector is compared to the input and the best matching neuron becomes winner in this computation step. Since we deal with time series, a typical stimulus consists of two parts:

1. the current input $\mathbf{s}^t$,
2. the context of the computation provided by the previous time step.

MSOM characterizes this context by the *merged content of the winner neuron in the previous time step*. Thus, a weight vector $(\mathbf{w}^i, \mathbf{c}^i) \in \mathbb{R}^n \times \mathbb{R}^n$ is attached to every neuron i, $\mathbf{w}^i$ representing the expected current stimulus, $\mathbf{c}^i$ representing the expected context. We fix a similarity measure d (e.g. the euclidean metric) for $\mathbb{R}^n$, a merge parameter $\gamma \in (0, 1)$, and a context weight $\alpha \in (0, 1)$. Then, the recurrent dynamic of the computation at time step t for sequence $(\mathbf{s}^1, \ldots, \mathbf{s}^t, \ldots)$ is determined by the following equation: the winner for time step t is

$$I(t) = \text{argmin}_i\{\tilde{d}_i(t)\}$$

where

$$\tilde{d}_i(t) = \alpha \cdot d(\mathbf{w}^i, \mathbf{s}^t) + (1 - \alpha) \cdot d(\mathbf{c}^i, C^t)$$

denotes the activation (distance) of neuron i, and C^t is the expected (merged) weight/context vector, i.e. the content of the winner of the previous time step

$$C^t = \gamma \cdot \mathbf{c}^{I(t-1)} + (1 - \gamma) \cdot \mathbf{w}^{I(t-1)} .$$

Thereby, the initial context C^0 is set to zero. Training takes place in Hebb style after each time step t:

$$\triangle \mathbf{w}^i = -\eta \cdot \text{nhd}(i, \mathbf{w}, t) \cdot \frac{\partial d(\mathbf{w}^i, \mathbf{s}^t)}{\partial \mathbf{w}^i} \text{ and } \triangle \mathbf{c}^i = -\eta \cdot \text{nhd}(i, \mathbf{w}, t) \cdot \frac{\partial d(\mathbf{c}^i, C^t)}{\partial \mathbf{c}^i}$$

where $\eta > 0$ is the learning rate and $\text{nhd}(i, \mathbf{w}, t)$ denotes the neighborhood range of neuron i. For standard SOM, this is a Gaussian shaped function of the distance from the winner $I(t)$ of i. For neural gas (NG), it is a Gaussian function which depends on the rank of neuron i if all neurons are ordered according to their distance from the current stimulus [19]. It is also possible to use non-euclidean lattices such as the topology of hyperbolic SOM [22].
Obviously, MSOM accounts for the temporal context by an explicit vector attached to each neuron which stores the preferred context of this neuron. The way in which the context is represented is crucial for the result, since the representation determines the induced similarity measure of sequences. Thus, two questions arise:

1. Which (explicit) similarity measure on sequences is induced by this choice?
2. What is the capacity of this model?

Interestingly, both questions can be answered for MSOM:

1. If neighborhood cooperation is neglected and provided enough neurons, Hebbian learning converges to the following stable fixed point of the dynamics:

   $$\mathbf{w}^{\text{opt}(t)} = \mathbf{s}^t, \mathbf{c}^{\text{opt}(t)} = \sum_{i=1}^{t-1} \gamma(1 - \gamma)^{i-1} \cdot \mathbf{s}^{t-i}$$

   and $\text{opt}(t)$ is winner for time step t [8].

2. If sequence entries are taken from a finite input alphabet, the capacity of MSOM is equivalent to finite state automata [28].

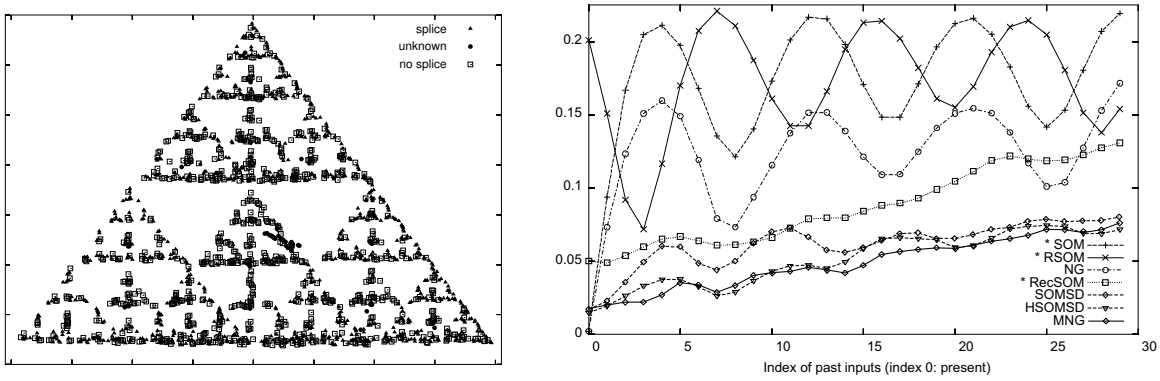

Figure 1: Left: Orthogonal projection of context weights $\mathbf{c}^i \in \mathbb{R}^3$ of 2048 prototypes trained with MNG on DNA-sequences. The shape of these points has strong similarity with images of fractals. Right: Temporal quantization error for different unsupervised models and the Mackey-Glass time series. Results marked with (*) are taken from [33].

Result (1) states that the representation of context which arises in the weights $\mathbf{c}^i$ consists of a leaky integration over the sequence entries, which is also known as a fractal encoding of the entries. This behavior can be demonstrated by inspection of the weight vectors which arise during training if the underlying source is fairly uniformly distributed, as shown in Fig. 1(left). The points depicted in this figure are obtained as context vectors if MSOM is trained using the NG neighborhood (MNG) and 2048 neurons for the DNA-sequences provided in [26]. Thereby, the letters T, C, G, A of the sequences are embedded as points into $\mathbb{R}^3$, and the merging parameter is set $\gamma = 0.5$. Interestingly, a posterior labeling of the prototypes into introns/non splice sites yields an error $< 15\%$ on the test set. (The situation is worse for exons for which only a short consensus string exists.) This is quite remarkable provided the fact that the training is done in an unsupervised way using simple Hebbian learning.

Result (2) gives an exact characterization of the capacity of MSOM networks in classical terms, similar to well-known results for supervised recurrent networks [21]. However, this fact does *not* tell us how to learn such an automaton based on given data. Because of (1), Hebbian learning might not be the best choice because of its focus on the most recent entries. It has been pointed out in [9], that Hebbian learning can be interpreted as a truncated gradient descent on an appropriate cost function in terms of the distances, the quantization error. Alternative learning schemes arise for different cost functions in terms of the distances and a different optimization strategy, e.g. a full gradient or EM approaches.

MSOM networks can be used for time series inspection or clustering since the map arranges the stimuli according to their recent context on the map taking the temporal statistics into account. A posterior labeling extends the application area to supervised time series classification or regression. This method is most suited for those tasks where the representation derived in (1) captures relevant information. This is the case e.g. for the speaker identification task based on the utterance of Japanese vowels provided in the UCI-KDD archive: 9 different speakers have to be identified based on the utterance of the Japanese vowel 'ae', represented by the cepstrum vectors with different stream length. MNG with posterior labeling allows to achieve a test error of 2.7% for 150 neurons and 1.6% for 1000 neurons (for details see [28]) which beats the error rate of 5.9% (rule based) resp. 3.8% (HMM) reported by the donator of the data set [16].

3 General recurrent networks

It has been pointed out in [9, 10] that several popular recurrent SOM models share their principled dynamics (up to minor details) whereby they differ in their internal representation of the context. In all cases, the context is extracted as the relevant part of the activation of the map in the previous step. Thereby, the notion of 'relevance' differs between the models. Assume an extraction function is fixed

$$\text{rep} : \mathbb{R}^N \to \mathbb{R}^r$$

where N is the number of neurons and r is the dimensionality of the context representation. Then the general dynamics is given by

$$\tilde{d}_i(t) = \alpha \cdot d(\mathbf{w}^i, \mathbf{s}^t) + (1-\alpha) \cdot d_r(\mathbf{c}^i, C^t)$$

where

$$C^t = \text{rep}\left(\tilde{d}_1(t-1), \ldots, \tilde{d}_N(t-1)\right)$$

extracts the relevant information from the activation of the previous time step, $\mathbf{c}^i \in \mathbb{R}^r$, and d_r is a similarity measure on $\mathbb{R}^r$. This formulation emphasizes the importance of an appropriate internal representation of complex signals by means of a context $\mathbf{c}^i$. The representation function rep extracts this information from the computation.
MSOM is obtained for $r = n$, $d_r = d$, and rep as the merged content of context and weight of the winner in the previous step. Alternative choices are reasonable (see [9]):

1. *Only the neuron itself*: the temporal Kohonen map (TKM) [3] performs leaky integration of the distances of each neuron. The dynamics can be obtained by setting $r = N$, rep = id, d_r as the standard dot product, and $\mathbf{c}^i$ as the i'th unit vector, which realizes the 'focus' of neuron i on its own activation. The recurrent SOM [15] is similar in spirit, but it integrates vectors instead of distances and requires a vectorial quantity $\tilde{d}_i(t)$.

2. *Full information:* the recursive SOM (RecSOM) [33] chooses $r = N$. $\text{rep}(x_1, \ldots, x_N) = (\exp(-x_1), \ldots, \exp(-x_N))$ is one-one, i.e. all information is kept. The feedback SOM is similar to RecSOM with respect to the context, however, the context integrates an additional leaky loop onto itself [11].

3. *Winner location:* the SOM for structured data (SOMSD) [6] is restricted to regular lattice structures. Denote by $L(i)$ the location of neuron i in a d-dimensional lattice. Then $r = d$ and $\text{rep}(x_1, \ldots, x_n) = L(i_0)$ where i_0 is the index of the winner $\text{argmin}_i\{x_i\}$. This context representation is only applicable to priorly fixed, though not necessarily euclidean lattices. SOMSD for a (fixed) hyperbolic lattice has been proposed in [29].

4. *Supervised recurrent networks:* share the dynamics; they result for $r = N$, $\text{rep}(x_1, \ldots, x_n) = (\text{sgd}(x_1), \ldots, \text{sgd}(x_N))$, and $d = d_r$ as the dot product. In this sense, the proposed dynamic is generic; it directly extends the dynamic of supervised recurrent networks.

For all settings, Hebbian learning can be applied, as discussed e.g. in [8]. Thereby, Hebbian learning can be interpreted as a truncated gradient descent on an appropriate cost function which depends on the distances and neighborhood structure of the map. The truncation disregards contributions back in time.

Obviously, these unsupervised recurrent methods separate into two categories: representation of the context in the data space as for TKM and MSOM, and representation of the context in a space which is related to the neurons as for SOMSD and RecSOM. In the latter case, the representation space can be enlarged if more neurons are considered. In the first case, the representation capability is restricted by the data dimensionality. We would like to point out that MSOM can be interpreted as the 'correct' implementation of TKM and RSOM regarding the following aspect: it has been pointed out in [32] that optimum weight representations of TKM and RSOM for a given sequence have the form $\mathbf{w}^{\mathrm{opt}(t)} = \sum_{i=0}^{t-1}(1-\alpha)^i \mathbf{s}^{t-i} / \sum_{i=0}^{t-1}(1-\alpha)^i$ which is quite similar to MSOM. However, there exist essential differences between the models; TKM does *not* converge to these optimum weights when using Hebbian learning. RSOM does because it uses a different learning rule, but the parameter α occurs in the encoding formula *and* in the dynamics. Usually, the dynamics is not very stable for large $(1-\alpha)$. Thus the encoding space cannot be utilized optimally by RSOM because instabilities of the dynamics would arise. MSOM allows to control these two parameters independently.
Apart from the encoding space, these methods differ with respect to several aspects: their memory and time *complexity* (RecSOM is quite demanding because of a large context dimensionality, MSOM is reasonable, SOMSD is cheap), the possibility to combine the approaches with alternative *lattices* (no restriction for MSOM and RecSOM, SOMSD requires a fixed prior lattice), and their principled *capacity*. This latter aspect is particularly interesting since it characterizes principled limits of these models. The following results have been achieved so far

1. TKM cannot represent all automata [28]. Thus it is strictly weaker than MSOM.
2. MSOMs are equivalent to finite automata, as already mentioned [28].
3. The same holds for SOMSD [8].
4. For RecSOM, the situation is difficult because of the quite complex context computation. On the one hand, an infinite reservoir is available because of real-valued context activations; on the other hand, however, information is very easily blurred because no focus in form of a winner computation takes place. The technical situation can be compared to the difficulties to investigate the capacity of supervised sigmoidal recurrent networks [14]. So far, it is known that
   (a) RecSOMs with small α can implement at most definite memory machines [30], i.e. focus on a finite time window.
   (b) RecSOMs with a slightly modified (i.e. normalized) context can simulate finite state automata [20].
   (c) RecSOMs with a simplified winner-takes-almost-all context (i.e. only the maximum values remain, the exponential function is substituted by a semilinear function) can simulate pushdown automata [20]. Pushdown automata are important for embedded constructions, e.g. embedded sentences in language processing.

Thus, there remain several open problems in particular for RecSOM. Note that these results do *not* state that these dynamics can be achieved using Hebbian learning, but they limit the principled capacity of the systems. In practice, the accuracy of the models depends on several factors including, in particular, the lattice topology and characteristics of the data (i.e. data topology and sparseness). The different capability of the models to adapt to temporal characteristics is exemplarily demonstrated in Fig. 1(right). The figure shows the

temporal quantization error as defined in [33] for the Mackey-Glass time series (see [33] for the description of the experiment) and two models without context (standard SOM and NG), the simple context of recurrent SOM, the full context provided by recursive SOM, SOMSD with rectangular and with hyperbolic (HSOMSD) lattice, and MSOM with data optimum NG context (MNG) (see [8] for details). Obviously, a different capability to learn the temporal structure arises, demonstrated by different quantization errors for past events.

4 Outlook on recursive networks for tree structures

We conclude with a remark on a generalization of the recurrent dynamic to tree structures. Binary trees or, more generally, trees with limited fan-out constitute good structures to represent data from interesting application areas such as chemistry, language parsing, bioinformatics, or image processing [27]. Supervised recurrent neural networks can be extended to so-called recursive neural networks by introducing more than one context vector. This principle is well established and it is accompanied by several successful applications and investigations of the models (see e.g. [7, 27] and references therein). A similar extension has been proposed for unsupervised models. For binary tree structures, two vectors $\mathbf{c}_1^i$ and $\mathbf{c}_s^i$ represent the context provided by the left and right subtree of a given vertex of the tree. Starting from the leafs of a tree, the distance of a tree v with label $l(v)$ and subtrees $t_1(v)$ and $t_2(v)$ can be determined by

$$\tilde{d}_i(v) = \alpha \cdot d\left(\mathbf{w}^i, l(v)\right) + (1-\alpha)\left(0.5 \cdot d_r\left(\mathbf{c}_1^i, C\left(t_1(v)\right)\right) + 0.5 \cdot d_r\left(\mathbf{c}_2^i, C\left(t_2(v)\right)\right)\right)$$

with context

$$C(t_1(v)) = \mathrm{rep}\left(\tilde{d}_1\left(t_1(v)\right), \ldots, \tilde{d}_N\left(t_2(v)\right)\right)$$

and analogous for t_2. Hebbian learning can be directly transferred to this setting. This principle has been tested for different data sets for SOMSD. A topological mapping of trees according to their structures and labels arises [6, 8, 27]. However, only preliminary results of the capacity of unsupervised models for tree structures have been presented so far [8, 9].

References

[1] Barreto, G., Araújo, A., and Kremer, S.C. (2003). A Taxonomy for Spatiotemporal Connectionist Networks Revisited: The Unsupervised Case. *Neural Computation*, 15(6):1255-1320.

[2] Bishop, C.M., Hinton, G.E., and Strachan, I.G.D. (1997a). *Proceedings IEE Fifth International Conference on Artificial Neural Networks*, Cambridge, U.K., 111-116.

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

[4] Euliano, N.R. and Principe, J.C. (1999). A spatiotemporal memory based on SOMs with activity diffusion. In E.Oja and S.Kaski (eds.), *Kohonen Maps*, Elsevier.

[5] Günter, S. and Bunke, H. (2002). Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23:401–417.

[6] Hagenbuchner, M., Sperduti, A., and Tsoi, A.C. (2003). A Self-Organizing Map for Adaptive Processing of Structured Data. *IEEE Transactions on Neural Networks* 14:191-505.

[7] Hammer, B. (2002). Recurrent networks for structured data - a unifying approach and its properties. *Cognitive Systems Research* 3(2), 145-165.

[8] Hammer, B., Micheli, A., Sperduti, A., Strickert, M. (2004). Recursive self-organizing network models. *Neural Networks* 17(8-9), 1061-1086.

[9] Hammer, B., Micheli, A., Sperduti, A., and Strickert, M. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing* 57, 3-35.

[10] Hammer, B., Micheli, A., and Sperduti, A. (2002). A general framework for unsupervised processing of structured data. In M.Verleysen (ed.), *European Symposium on Artificial Neural Networks*, pages 389-394, D-side publications.

[11] Horio, K. and Yamakawa, T. (2001). Feedback self-organizing map and its application to spatio-temporal pattern classification. *International Journal of Computational Intelligence and Applications* 1(1):1-18.

[12] James, D.L. and Miikkulainen, R. (1995). SARDNET: a self-organizing feature map for sequences. In G.Tesauro, D.Touretzky, and T.Leen (eds.), *Advances in Neural Information Processing Systems 7*, pages 577-584, MIT Press.

[13] Kaski, S., Kangas, J., and Kohonen, T. (1998). Bibliography of self-organizing maps, papers: 1981-1997. *Neural Computing Surveys*. 1, 102-350.

[14] Kilian, J. and Siegelmann, H.T. (1996). The dynamic universality of sigmoidal neural networks. *Information and Computation*, 128.

[15] Koskela, T., Varsta, M., Heikkonen, J., and Kaski, K. (1998). Time Series Prediction using Recurrent SOM with Local Linear Models. *Int. J. of Knowledge-Based Intelligent Engineering Systems* 2(1): 60-68.

[16] Kudo, M., Toyama, J., and Shimbo, M. (1999). Multidimensional Curve Classification Using Passing-Through Regions. *Pattern Recognition Letters* 20(11–13), 1103–1111.

[17] Kohonen, T. (1997). *Self-organizing maps*. Springer.

[18] T. Kohonen and P. Somervuo (2002), How to make large self-organizing maps for nonvectorial data, *Neural Networks* **15:**945-952.

[19] Martinetz, T., Berkovich, S., and Schulten, K. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE-Transactions on Neural Networks* 4(4): 558-569.

[20] Neubauer, N. (2005). Recursive SOMs and Automata. M.Sc. Thesis, Cognitive Science, University of Osnabrück.

[21] Omlin, C.W., and Giles, C.L. (1996). Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 43(6):937-972.

[22] Ritter, H. (1999). Self-organizing maps in non-euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–108. Springer.

[23] Schulz, R. and Reggia, J.A. (2004). Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps. *Neural Computation* 16(3): 535–561.

[24] Simon, G., Lendasse, A., Cottrell, M., Fort, J.-C., and Verleysen, M. (2003). Double SOM for Long-term Time Series Prediction *WSOM 2003, Workshop on Self-Organizing Maps*, Hibikino (Japan), 11-14 September 2003, pp. 35-40.

[25] Somervuo, P.J. (2004). Online algorithm for the self-organizing map of symbol strings. *Neural Networks*, 17(8-9):1231-1240.

[26] Sonnenburg, S. (2002). New methods for splice site recognition. Diploma Thesis, Institut für Informatik, Humboldt-Universität Berlin.

[27] Sperduti, A. (2001). Neural networks for adaptive processing of structured data. In: G.Dorffner, H.Bischof, K.Hornik (eds.), *ICANN'2001*, pages 5-12, Springer.

[28] Strickert, M. and Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing* 64:39-72, 2005

[29] Strickert, M., Hammer, B., and Blohm, S. (2005). Unsupervised recursive sequences processing. *Neurocomputing* 63, 69-98, 2005.

[30] Tino, P., Farkas, I., and van Mourik, J. (2005). Topographic Organization of Receptive Fields in Recursive Self-Organizing Map. Technical Report CSRP-05-06, School of Computer Science, University of Birmingham, UK.

[31] Tino, T., Kaban, A., and Sun, Y. (2004). A Generative Probabilistic Approach to Visualizing Sets of Symbolic Sequences. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - (KDD-2004)*, R. Kohavi, J. Gehrke, W. DuMouchel, J. Ghosh (eds). pp. 701-706, ACM Press.

[32] Varsta, M., Heikkonen, J., Lampinen, J., and Milán, J.del R. (2001). Temporal Kohonen map and recurrent self-organizing map: analytical and experimental comparison. *Neural Processing Letters*, 13(3):237-251.

[33] Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks* 15(8-9):979-992.

[34] Wiemer, J.C. (2003). The time-organized map algorithm: extending the self-organizing map to spatiotemporal signals. *Neural Computation* 15(5): 1143–1171.