# Merge SOM for temporal data

Marc Strickert

*Pattern Recognition Group,*
*Institute of Plant Genetics and Crop Plant Research Gatersleben*

Barbara Hammer

*Institute of Computer Science,*
*Technical University of Clausthal*

**Abstract**

The recent merging self-organizing map (MSOM) for unsupervised sequence processing constitutes a fast, intuitive, and powerful unsupervised learning model. In this paper, we investigate its theoretical and practical properties. Particular focus is put on the context established by the self-organizing MSOM, and theoretic results on the representation capabilities and the MSOM training dynamic are presented. For practical studies, the context model is combined with the neural gas vector quantizer to obtain merging neural gas (MNG) for temporal data. The suitability of MNG is demonstrated by experiments with artificial and real-world sequences with one- and multi-dimensional inputs from discrete and continuous domains.

*Key words:* Self-organizing map, temporal Kohonen map, sequence processing, recurrent models, neural gas, fractal encoding, finite automata, speaker identification, DNA splice site detection.

## 1 Introduction

The design of recursive data models is a challenging task for dealing with structured data. Sequences, as a special case, are given by series of temporally or spatially connected observations – DNA chains and articulatory time series are two interesting examples of sequential data. Sequential data play a major role in human processing of biological signals since virtually all visual,

*Email addresses:* stricker@ipk-gatersleben.de (Marc Strickert), hammer@in.tu-clausthal.de (Barbara Hammer).

auditory, or sensorimotor data are given as sequences of time-varying stimuli. Thus, biologically plausible and powerful neural sequence processing models are interesting from a practical as well as from a cognitive point of view.

Unsupervised self-organization is a particularly suitable paradigm for biological learning because it does not rely on explicit teacher information and as it uses intuitive primitives like neural competition and cooperation. Kohonen's self-organizing map (SOM) is a well-known method for projecting vectors of a fixed, usually high dimension onto a low-dimensional grid, this way enabling analysis and visualization [11]. In contrast to the SOM, the neural gas (NG) algorithm yields representations based on prototypes in the data space, which cooperate in a data-driven topology and yield small quantization errors [13]. However, in their original formulations, SOM and NG have been proposed for processing real-valued vectors, not for sequences.

Given sequential data, vector representations usually exist or can be found for single sequence entries. For vector quantizer models, such as the mentioned SOM and NG, temporal or spatial contexts within data series are often realized by means of data windows. These windows are constructed as serialized concatenations of a fixed number of entries from the input stream; problems related to this are a loss of information, the curse of dimensionality, and usually inappropriate metrics. The last issue, inappropriateness, can be tackled only partially by using adaptive metrics [17,19].

Recently, increasing interest can be observed in unsupervised recurrent self-organizing networks for straight sequence processing. Prominent methods are the temporal Kohonen map (TKM), the recurrent self-organizing map (RSOM), recursive SOM (RecSOM), and SOM for structured data (SOMSD) [1,7,24,26]. These are unsupervised neural sequence processors with recurrent connections for recursive element comparisons. Thereby, similarity structures of entire sequences emerge as result of recursive operations. TKM and RSOM are based on biologically plausible leaky integration of activations, and they can be used to explain biological phenomena such as the development of direction selectivity in the visual cortex [4]. However, these two models use only a local form of recurrence. RecSOM and SOMSD use a richer recurrence as demonstrated in several experiments [7,26].

First experimental comparisons of the methods and steps towards an investigation of their theoretical properties have been presented recently in the articles [8,9]. It turns out that all models obey the same recurrent winner dynamic, but they differ in the notion of context; internally, they store sequences in different ways. This has consequences on their efficiency, their representation capabilities, the induced metrics, and the possibility to combine their contexts with different lattice structure. The focus of this work is an investigation of the recent merge SOM (MSOM) approach [20]. MSOM is based upon

a very effective context model which can be combined with arbitrary lattice structures: the temporal context of MSOM combines the currently presented pattern with the sequence history in an intuitive way by referring to a merged form of the winner neuron's properties.

Especially for unsupervised approaches, a thorough understanding of the emerging representation is needed for a valid interpretation. In this article, we will investigate the representation model and the capacity of MSOM in theory and practice. We demonstrate that MSOM develops a kind of fractal sequence encoding which makes it comparable to the context representation of TKM and RSOM. However, unlike TKM, the encoding is a stable fixed point of the learning dynamic, and it has got a larger flexibility than RSOM, because the internal representation and the influence of the context can be controlled separately. We prove that MSOM, unlike TKM and RSOM, can simulate finite automata with constant delay. Moreover, we demonstrate in several experiments the appropriateness of the model for unsupervised and supervised learning tasks.

## 2 Related Work

Before the MSOM context model is discussed in detail, a quick reminder of the original self-organizing map and alternative vector quantization models is given. Then the focus is put on the short revision of existing SOM-based models for sequences before introducing MSOM.

### *2.1 Models for plain vector processing*

The original self-organizing map for plain vector processing is given by a set of neurons $\mathcal{N} = \{1, \ldots, m\}$ equipped with weights $\boldsymbol{w}^i \in \mathbb{R}^n$, and a topology of the neuron configuration with a function $d_{\mathcal{N}} : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$. In the standard SOM, neurons are usually connected in a two-dimensional grid, and $d_{\mathcal{N}}$ denotes the distance of two neurons in that grid. A similarity measure on the weight space $\mathbb{R}^n$ is fixed; often the standard squared Euclidean metric $\|\cdot\|^2$ is taken. The goal of training is to find prototype weights which represent given data points well in $\mathbb{R}^n$; simultaneously, the topology of the neural grid should match the topology of the data. For training, data vectors are iteratively presented, and the weights of the closest neuron and its neighbors in the grid are adapted towards the currently processed data point. The update formula for neuron $i$ given pattern $\mathbf{x}^j$ is given by the formula

$$\Delta \boldsymbol{w}^i = \gamma \cdot \mathrm{h}_\sigma(d_{\mathcal{N}}(i, I)) \cdot (\mathbf{x}^j - \boldsymbol{w}^i) \,,$$

where $I$ is the winner, i.e. the neuron with smallest distance from $\mathbf{x}^j$, and $\mathrm{h}_\sigma(\cdot)$ is a decreasing function of increasing neighborhood distance, e.g. a Gaussian bell-shaped curve or a derivative of the Gaussian.

Popular alternatives to the standard SOM with different target topologies are the neural gas (NG) and the hyperbolic SOM (HSOM). The HSOM model operates on a target lattice with exponentially increasing neighborhood size [16]; NG adapts all neurons according to their rank with respect to the distance from the current data point, i.e. according to a data-driven optimal lattice structure without topology constraints [13].

### *2.2 Models for sequence processing*

A number of recursive modifications of the SOM have been proposed in literature, and the most prominent models are briefly revisited for a motivation of the MSOM model.

**TKM**, the temporal Kohonen map, is an early approach to SOM-based sequence processing which has been proposed by Chappell and Taylor [1]. The TKM neurons implement recurrence in terms of leaky signal integration. For a sequence $(\mathbf{x}^1, \ldots, \mathbf{x}^t)$ with entries $\mathbf{x}^i \in \mathbb{R}^n$, the integrated distance of neuron $i$ with weight vector $\boldsymbol{w}^i$ is computed by

$$\mathrm{d}_i(t) = \sum_{j=1}^{t} \alpha \cdot (1-\alpha)^{(t-j)} \cdot \|\mathbf{x}^j - \boldsymbol{w}^i\|^2 \,.$$

This distance expresses the neurons' quality of fit, not only to the current input pattern, but also to the exponentially weighted past. The parameter $\alpha \in (0;1)$, kept constant, controls the rate of signal decay during summation. This way, the balance between the currently processed and the historic inputs affects winner selection for getting the best matching compromise between the present and the past.

An equivalent recursive formulation for the integrated distance $\mathrm{d}_i(t)$ of neuron $i$ after the $t$th time step is $\mathrm{d}_i(t) = \alpha \cdot \|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + (1-\alpha) \cdot \mathrm{d}_i(t-1)$ with $\mathrm{d}_i(0) := 0$. TKM training is realized by adapting the weights $\boldsymbol{w}^i$ at each time step towards the unintegrated current input $\mathbf{x}^t$ using the standard SOM update rule $\Delta\boldsymbol{w}^i = \gamma \cdot \mathrm{h}_\sigma(d_{\mathcal{N}}(i, I)) \cdot (\mathbf{x}^t - \boldsymbol{w}^i)$ [1]. This way, by making the prototype weights more similar to the input vector, the correlation of either is increased which is known as Hebbian learning. Temporal context for TKM is represented by the activation of the winner neuron in the previous time step. Evidently, this context choice does not depend on the particular choice of the neuron target grid architecture and it can be combined with hyperbolic lattices or the data-optimum topology of neural gas.

**RSOM**, the recurrent SOM proposed by Koskela et al. is similar to TKM; it takes the integrated direction of change into account, not only for distance calculation but also for weight update [12]. Both RSOM and TKM refer to context by signal integration. Only RSOM stores context information in the weight vectors for further comparison. Due to vector summation, the distinction between the currently processed pattern and the history is impossible, which makes the interpretation and analysis of trained RSOMs a difficult task.

**RecSOM** uses a richer context representation [26]: in addition to a weight $\boldsymbol{w}^i \in \mathbb{R}^n$, each neuron $i$ possesses a vector $\boldsymbol{c}^i \in \mathbb{R}^{|\mathcal{N}|}$ representing the temporal context as the activation of the entire map in the previous time step. The distance is

$$\mathrm{d}_i(t) = \alpha \cdot \|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + \beta \cdot \|\mathsf{c}^t - \boldsymbol{c}^i\|^2 \,,$$

whereby $\alpha$, $\beta > 0$ are fixed values to control the influence of the context compared to the presented sequence element. The current context describing the previous time step is an exponentially weighted vector of neuron activations in the last time step, $\mathsf{c}^t := (\exp(-\mathrm{d}_1(t-1)), \ldots, \exp(-\mathrm{d}_{|\mathcal{N}|}(t-1)))$. Training adjusts both weight and context of the winner neuron and its neighborhood towards the current values $\mathbf{x}^t$ and $\mathsf{c}^t$. This context type does not depend on a regular neuron grid and it can be used also with hyperbolic lattices or with NG-topology. However, this context model is computationally costly because extra dimensions related to the number $|\mathcal{N}|$ of neurons in the map are added to each neuron.

**SOMSD** has been proposed for general tree structures [7], but we focus on the special case of sequences. Like for RecSOM, an additional context vector $\boldsymbol{c}^i$ is used for each neuron, but only the last winner index is stored as lightweight information about the previous map state. Thus, $\boldsymbol{c}^i$ is element of a real vector space which includes the grid of neurons, e.g. $\boldsymbol{c}^i \in \mathbb{R}^2$, if a two-dimensional rectangular lattice is used. SOMSD distances are

$$\mathrm{d}_i(t) = \alpha \cdot \|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + \beta \cdot d_G(I_{t-1}, \boldsymbol{c}^i) \,,$$

whereby $I_{t-1}$ denotes the index of the winner in step $t-1$. $d_G$ measures distances in the grid, usually in terms of the squared Euclidean distance of the neurons' low-dimensional lattice coordinates. During training, the winner and its neighborhood adapt weights and contexts of $\mathbf{x}^t$ and $I_{t-1}$, respectively. Thereby, grid topology and the indices of neurons are expressed as elements of a real-vector space. Obviously, the context of SOMSD relies on ordered addressing in terms of the topology of the neural map; consequently, it cannot be combined with alternative models such as neural gas. If complex sequences are mapped to low-dimensional standard Euclidean lattices, topological mismatches can be expected. Thus, since the distance between context vectors is defined by the lattice-dependent measure $d_G$, this approach must be carefully revised to handle alternative target lattices.

**HSOM-S** transfers the idea of SOMSD to more general grids expressed by a triangulation of a two-dimensional manifold, such as a hyperbolic plane, in order to better match the data topology of sequences, as discussed in [21]. The previous winner, adjusted through Hebbian learning, is referred to by a coordinate within a triangle of adjacent neurons. For a stable learning, the context influence starts at $\beta = 0$, for getting an initially weight-driven SOM ordering, and it is then increased to a desired influence value. Such an approach constitutes a step towards more general lattice structures which are better suited for the presentation of input sequences but it is still limited to a priorly fixed topology.

Obviously, RecSOM offers the richest notion of history because temporal context is represented by the activation of the entire map in the previous time step; therefore, however, RecSOM also constitutes the computationally most demanding model. SOMSD and HSOM-S still use global map information, but in a compressed form; their storage of only the location of the winner is much more efficient and noise-tolerant, albeit somehow lossy, in comparison to the whole activity profile. In both models, context similarity, expressed by the comparison of distances between grid locations, depends on the priorly chosen static grid topology. For this reason, this back-referring context model cannot be combined with the dynamic rank-based topology of NG for data-optimal representations. Context of TKM and RSOM is just implicitly represented by the weights of neurons, i.e. without explicit back-reference to previous map activity; hence, both methods' sequence representation domains are restricted to superpositions of values from the domain of the processed sequence entries.

A brief overview of context complexity representable by the models is given in [20], a more detailed discussion is found in [9]. With reference to the properties of the above-mentioned models, a wish list for a new type of context model is derived: The lesson learned from Voegtlin's RecSOM is the beneficial consideration of the map activity in the previous time step as independent part of the neurons. SOMSD and HSOM-S compress the costly RecSOM activation profile to only the topological index of the last winner in the neuron grid. NG yields better quantization performance than the SOM. TKM context is given by signal integration; this element superposition would yield efficient fractal context encodings, if these were stable fixed points of the TKM training dynamic, but usually they are not. In the following, we define and investigate in detail another model, the merge SOM (MSOM), which allows to integrate ideas from the listed methods and to also tackle some of the mentioned deficiencies.

The MSOM model has been proposed in the article [20] and first theory and experiments have been presented in [9,22]. MSOM combines a noise-tolerant learning architecture which implements a compact back-reference to the previous winner with separately controllable contribution of the current input and the past with arbitrary lattice topologies such as NG. It converges to

an efficient fractal encoding of given sequences with high temporal quantization accuracy. With respect to representation, MSOM can be interpreted as an alternative implementation of the encoding scheme of TKM, but MSOM possesses larger flexibility and capacity due to the explicit representation of context: In MSOM neurons specialize on both input data and previous winners in such a way that neuron activation orders become established; thus, order is coded by a recursive self-superposition of already trained neurons and the current input. Consequently, this yields a successive refinement of temporal specialization and branching with fractal characteristics similar to Lindenmayer's L-systems. Unlike RecSOM, this context representation is space-efficient and unlike SOMSD it can be combined with arbitrary lattices.

## 3 The Merge SOM Context (MSOM)

In general, the merge SOM context refers to a fusion of two properties characterizing the previous winner: the weight and the context of the last winner neuron are merged by a weighted linear combination. During MSOM training, this context descriptor is kept up-to-date and it is the target for the context vector $\boldsymbol{c}^j$ of the winner neuron $j$ and its neighborhood. Target means that the vector tuple $(\boldsymbol{w}^i, \boldsymbol{c}^i) \in \mathbb{R}^{2n}$ of neuron $i$ is adapted into the direction of the current pattern and context according to Hebbian learning.

### *3.1 Definition of the Merge Context*

A MSOM network is given by neurons $\mathcal{N} = \{1, \ldots, m\}$ which are equipped with a weight $\boldsymbol{w}^i \in \mathbb{R}^d$ and context $\boldsymbol{c}^i \in \mathbb{R}^d$. Given a sequence entry $\mathbf{x}^t$, the winner $I_t$ is the best matching neuron $i$, for which its recursive distance

$$\mathrm{d}_i(t) = (1-\alpha) \cdot \|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + \alpha \cdot \|\mathsf{c}^t - \boldsymbol{c}^i\|^2$$

to the current sequence entry $\mathbf{x}^t$ and the context descriptor $\mathsf{c}^t$ is minimum. Both contributions are balanced by the parameter $\alpha$. The context descriptor

$$\mathsf{c}^t = (1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}}$$

is the linear combination of the properties of winner $I_{t-1}$ in the last time step, setting $\mathsf{c}^1$ to a fixed vector, e.g. $\mathbf{0}$. A typical merging value for $0 \leq \beta \leq 1$ is $\beta = 0.5$.

Training takes place by adapting both weight and context vector towards the current input and context descriptor according to the distance of the neuron

from the winner given by a neighborhood function $d_{\mathcal{N}} : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ that defines the topology of MSOM neurons:

$$\Delta \boldsymbol{w}^i = \gamma_1 \cdot \mathrm{h}_\sigma(d_{\mathcal{N}}(i, I_t)) \cdot (\mathbf{x}^t - \boldsymbol{w}^i)$$
$$\Delta \boldsymbol{c}^i = \gamma_2 \cdot \mathrm{h}_\sigma(d_{\mathcal{N}}(i, I_t)) \cdot (\mathsf{c}^t - \boldsymbol{c}^i)$$

$\gamma_1$ and $\gamma_2$ are the learning rates and, as beforehand, $\mathrm{h}_\sigma$ is usually a Gaussian shaped function.

### *3.2 Merge Context for Neural Gas (MNG)*

Since the context does not depend on the lattice topology, a combination of the dynamic with alternative lattice structures like the neural gas model [13] or the learning vector quantization model [11] is easily possible. Here, we combine the context model with neural gas to obtain merging neural gas (MNG).

The recursive winner computation is the same as for MSOM. The adaptation scheme is customized appropriately to incorporate the data-driven neighborhood. After presenting a sequence element $\mathbf{x}^t$, the rank

$$k = \mathrm{rnk}(i) = \left| \{ i' \mid \mathrm{d}_{i'}(t) < \mathrm{d}_i(t) \} \right|$$

of each neuron $i$ is computed. This accounts for the topological information that a number of $k$ neurons is closer to $\mathbf{x}^t$ than neuron $i$ is. Upon that, rigid neuron competition is established, because the update amount is an exponential function of the rank:

$$\Delta \boldsymbol{w}^i = \gamma_1 \cdot \exp(-\mathrm{rnk}(i)/\sigma) \cdot (\mathbf{x}^t - \boldsymbol{w}^i)$$
$$\Delta \boldsymbol{c}^i = \gamma_2 \cdot \exp(-\mathrm{rnk}(i)/\sigma) \cdot (\mathsf{c}^t - \boldsymbol{c}^i)$$

The context descriptor $\mathsf{c}^t$ has to be updated during training to keep track of the respective last winner. In experiments, the learning rates were set to identical values $\gamma_1 = \gamma_2 = \gamma$. The neighborhood influence $\sigma$ decreases exponentially during training to get neuron specialization. The two update rules minimize separately the average distortion error between the internal prototype states and the target points in terms of a stochastic gradient descent [13]. In combination, these separate minima need not lead to a global minimum; however, the minimization target of either can be controlled via the distance measure $\mathrm{d}_i(t)$.

*Choice of the weight/context balance parameter $\alpha$*

A crucial parameter of the recursive distance $\mathrm{d}_i(t)$ is $\alpha$ which is considered in this paragraph. The initial contribution of the context term to the distance computation and thus to the ranking order is chosen low by setting the weight/context balance parameter $\alpha$ to a small positive value. Since the weight representations become more reliable during training, gradually more attention can be paid to the specific contexts that refer to them. Thus, after an initial weight specialization phase, $\alpha$ can be released and directed to a value that maximizes the neuron activation entropy.

In computer implementations, such an entropy-controlled adaptation has been realized: $\alpha$ is increased, if the entropy is decreasing, i.e. the representation space for the winner calculation is widened by allowing more context influence to counteract the specialization of the neurons on only the input patterns; otherwise, if the entropy is increasing, $\alpha$ is slightly decreased in order to allow a fine tuning of the context influence and of the ordering. As a result, the highest possible number of neurons should — on average — be equally active by the end of training. This entropy-driven $\alpha$-control strategy has proved to be very suitable for obtaining good results in the experiments.

### *3.3 Properties of the merge context*

For MNG and MSOM learning, the optimum choice for the weight vector and the context vector is obtained for their minimum distance to the currently presented pattern and the context descriptor. It has been shown for vanishing neighborhoods, i.e. for the finalizing training phase, that the optimum weight vector converges towards the pattern which is responsible for winner selection and that the optimum context vector converges to a fractal encoding of the previously presented sequence elements [9]. The term 'fractal' is used to express that ordered temporal element superposition leads to a more or less uniform filling of the state space, this way relating location to meaning. Importantly, the optimum vectors for fractal encoding result as stable fixed points from the MSOM or MNG dynamic.

**Theorem 1** *Assume that MSOM with recursive winner computation*

$$\mathrm{d}_i(t) = (1-\alpha) \cdot \|\boldsymbol{w}^i - \mathbf{x}^t\|^2 + \alpha \cdot \|\mathsf{c}^i - \left((1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}}\right)\|^2$$

*is given. Assume the initial context is* $\mathsf{c}^1 := \mathbf{0}$*. Assume a sequence with entries* $\mathbf{x}^j$ *for* $j \geq 1$ *is presented. Then the winner's optimum weight and context*

*vectors at time $t$*

$$\boldsymbol{w}^{\mathrm{opt}(t)} = \mathbf{x}^t, \quad \boldsymbol{c}^{\mathrm{opt}(t)} = \sum_{j=1}^{t-1} \beta \cdot (1-\beta)^{j-1} \cdot \mathbf{x}^{t-j} \,,$$

*emerge from the learning dynamic as stable fixed points, provided that enough neurons are available, that neighborhood cooperation is neglected, and that the vectors $\sum_{j=1}^{t} \beta \cdot (1-\beta)^{j-1} \cdot \mathbf{x}^{t-j}$ are pairwise different for each $t$.*

For convenience, a proof of Theorem 1 is provided in the Appendix A; this proof is a simplified version of the one given in [9]. One can see from the appended proof that, starting from random positions, weights and contexts converge to the postulated settings in the late training phase, when neighborhood cooperation vanishes. In addition, the proof supports the experimental findings that convergence takes place consecutively, first for the weights, then for the context representations further back in time. This fact may explain the success of entropy-based choices of $\alpha$, which guarantees the successive convergence of weights and contexts over time.

In subsequent experiments, the emergence of a space-filling set resembling the fractal Cantor set with non-overlapping context can be observed for a merging parameter of $\beta = 0.5$. The spreading dynamic of the zero-initialized context in the weight space is self-organizing with respect to the density of the contextual input, which is a benefit over encoding ab initio. Since the context is a function of the previous winner's weight and context, the adaptation is a moving target problem; therefore, it is generally a good policy to update weights faster than contexts, or to put more influence on the pattern matching than on the context matching by choosing $\alpha < 0.5$.

The theoretical result of the Theorem 1 neglects the neighborhood structure of the neural map which is responsible for topological ordering of the weights and contexts. Theoretical investigations of neighborhood cooperation for MNG and MSOM face severe problems: unlike NG, the MNG update dynamic is, due to context integration, only an approximate gradient descent of an appropriate error function as shown in [8]. Therefore, a potential function is not available for this case. For MSOM, the investigation of both correlated components, the weight and context representation, would be necessary. Also for standard SOM, results are available only for the one-dimensional case and results for higher dimensions are restricted to factorizing distributions [3,10]. Thus, these techniques cannot be transferred to our setting. However, in the experiment section of this paper, we will provide experimental evidence that a reasonable ordering of both, weights and contexts, can be observed for MNG, the MSOM model combined with neural gas.

We would like to mention that TKM and RSOM are also based on a fractal encoding of sequences, as pointed out in the articles [12,24,25]. TKM uses weight

vectors only for which the standard Hebbian learning does not yield the optimum codes as fixed points of the training dynamic. RSOM uses a different update scheme which is based on integrated distances and which yields optimum codes during training. Unlike MSOM, no explicit context model is used. Therefore, a parameter $\alpha$ which controls the influence of the context for winner determination and a parameter $\beta$ which controls the internal representation of sequences cannot be optimized independently. Instead, the single context parameter $\alpha$ determines both the context influence on the winner selection and the adaptation of the sequence representation space. MSOM allows an independent optimization of these parameters. In particular, as explained above, it allows a very efficient adaptation scheme for the parameter $\alpha$ based on the map entropy. We will see in experiments that values $\alpha \neq \beta$ are beneficial in practical applications.

An additional advantage of MSOM is that, by dint of the winner computation, information of longer time spans than for TKM and RSOM can be stored, and MSOM is less affected by noise. This claim can be accompanied by an exact mathematical result: the dynamic of MSOM allows to simulate finite automata with constant delay whereas the dynamic of TKM does not. We now address this fundamental issue, the representation capacity of MSOM in principle. This refers to the question whether one can explicitly characterize all computations possible within the dynamic which can be implemented by the recursive winner computation defined above. Thereby, aspects of learning or topological ordering are neglected and, as beforehand, we disregard neighborhood cooperation. In particular, the lattice topology is not relevant. For simplicity, we assume a finite input alphabet.

We will show that it is in fact possible to find an equivalent characterization of the capacity in these terms: the recursive winner dynamic based on the merge context is equivalent to finite automata, as specified and proved below. We will also show that TKM cannot simulate automata with constant delay. This provides a mathematical proof that the more general recursion of MNG is strictly more powerful than TKM.

We first recall the definition of automata and then we define the notion of simulation of an automaton by a recursive self-organizing map model with constant delay. Afterwards, we turn towards the results describing the capacity of MSOM and TKM.

**Definition 1** *A* finite automaton *consists of a triple*

$$A = (\Sigma, S, \delta)$$

*whereby* $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ *is a finite alphabet of input symbols,* $S = \{\mathrm{sta}_1, \ldots, \mathrm{sta}_{|S|}\}$ *is a set of states, and* $\delta : S \times \Sigma \to S$ *is the transition function.* $\Sigma^*$ *denotes the set of finite sequences over* $\Sigma$. *The automaton* $A$ accepts *a*

*sequence* $s = (\sigma_{i_1}, \ldots, \sigma_{i_t}) \in \Sigma^*$ *iff* $\delta^{it}(s) = \mathrm{sta}_{|S|}$*, whereby* $\delta^{it}$ *is defined as the recursive application of the transition function* $\delta$ *to elements of* $s$*, starting at the initial state* $\mathrm{sta}_1$*:*

$$\delta^{it}(s) = \begin{cases} \mathrm{sta}_1 & \text{if } s \text{ is the empty sequence, i.e. } t = 0, \\ \delta(\delta^{it}(\sigma_{i_1}, \ldots, \sigma_{i_{t-1}}), \sigma_{i_t}) & \text{if } t > 0\,. \end{cases}$$

The dynamic equation of MSOM and TKM has been defined for all *entries* of one given sequence. In a natural way, we extend this notation to the case of different *input sequences*: given a sequence $s = (\mathbf{x}^1, \ldots, \mathbf{x}^t)$ with elements in $\mathbb{R}^d$, the distance of $s$ from neuron $i$ is defined as

$$\mathrm{d}_i(s) = \mathrm{d}_i(\mathbf{x}^t)\,,$$

i.e. it yields the distance of the last element $\mathbf{x}^t$ of the sequence $s$ which is processed within the context given by the previous entries of $s$. This extension allows to introduce the notation of simulation of an automaton with constant delay by a recursive neural map.

**Definition 2** *Assume a given finite automaton* $A = (\Sigma, S, \delta)$*. A recursive self-organizing map with input sequences in* $(\mathbb{R}^d)^*$ simulates $A$ with constant delay `delay` *if the following holds: there exists a mapping* $\mathrm{enc} : \Sigma \to (\mathbb{R}^d)^{\mathtt{delay}}$ *and a specified neuron with index* $I_0$ *such that for every sequence* $s$ *in* $\Sigma^*$ *holds:*

$$s \text{ is accepted by } A \iff I_0 = \mathrm{argmin}_j\{\mathrm{d}_j(\mathrm{enc}(s))\}\,,$$

*whereby* $\mathrm{enc}(s)$ *denotes the component-wise application of* enc *to* $s = (\sigma_{i_1}, \ldots, \sigma_{i_t})$*:* $(\mathrm{enc}(\sigma_{i_1}), \ldots, \mathrm{enc}(\sigma_{i_t}))$*.*

The dynamic provided by the merge context can simulate finite automata with constant delay. More precisely, the following holds:

**Theorem 2** *Assume that* $A = (\Sigma, S, \delta)$ *is a finite automaton with alphabet* $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ *and states* $S = \{\mathrm{sta}_1, \ldots, \mathrm{sta}_{|S|}\}$*. Then a neural map with merge context and* $L_1$*-norm can be found with input dimension* $d = |S| \cdot |\Sigma| + 4$*,* $m = \mathcal{O}(|S| \cdot |\Sigma|)$ *neurons, and initial context* $\mathtt{c}^0$*, and an encoding function* $\mathrm{enc} : \Sigma \to (\mathbb{R}^d)^4$ *exists which depends only on the quantities* $|S|$ *and* $|\Sigma|$*, such that the neural map simulates* $A$ *with delay* 4*.*

The proof can be found in the Appendix B.

Since, when the merge context is used, a fixed map only yields a finite number of different internal states, it is obvious that a recursive map provided with the merge context can simulate at most finite automata. Thus, equivalence holds.

In contrast, the TKM cannot simulate every automaton with a constant delay which is independent of the automaton because of its restricted context representation. More precisely, one can show the following result:

**Theorem 3** *Assume that the alphabet $\Sigma$ consists of only one element $\sigma$. Assume a fixed constant delay. Assume a finite automaton with alphabet $\Sigma$ being simulated by a given TKM and an encoding function with constant delay. Then the automaton is trivial, i.e. it accepts all sequences or it rejects all sequences.*

The proof is provided in the Appendix C.

## 4 Experiments

In the following, experiments are presented to shed light on different aspects of the MNG model, such as the quality of the obtained context representation, the storage lengths of learned histories, the reconstruction of temporal information from trained networks, and the potential for using posterior neuron labeling for classification tasks.

Six data sets have been used for training: three discrete and three continuous sets. In the discrete domain, $\langle 1 \rangle$ a stream of elements from binary automata, $\langle 2 \rangle$ words generated with the Reber grammar, and $\langle 3 \rangle$ DNA sequences are processed; in the continuous domain, $\langle 4 \rangle$ the continuous one-dimensional Mackey-Glass time series, $\langle 5 \rangle$ a medical three-variate series with physiological observations, and $\langle 6 \rangle$ speaker articulation data containing sequences of 12-dimensional cepstrum vectors.

Thereby, datasets $\langle 3 \rangle$ and $\langle 6 \rangle$ also contain class information which can be used to evaluate the map formation with respect to this auxiliary data. The DNA sequences will be considered for splice site detection, and the articulation data will be used for speaker identification; the classifications are obtained from the map by calculating the activation histograms of neurons for different data to get a probability-based class assignment to the neurons.

### *4.1 Binary automata*

In the first learning task, the focus is put on the representation of the most likely sub-words in binary sequences generated by automata with given transition probabilities. Two questions concerning the MNG context are addressed: Can the fractal encoding stated in theory be observed in experiments? What is the empirical representation capability?

*Context development*

Figure 1 displays the experimental context space resulting from MNG training of 128 neurons for a random binary sequence that contains $10^6$ symbols of **0** and **1** independently drawn with $P(\mathbf{0}) = P(\mathbf{1}) = 1/2$. Parameters are a learning rate of $\gamma = 0.03$, a fair combination of context and weight by $\beta = 0.5$, and an initial context influence of $\alpha = 0.001$ that has reached — by the entropy-based parameter control — a final value of $\alpha = 0.34$ after training. During adaptation, two of the 128 neurons have fallen idle, because the specialization on context, being subject to further updates, is a moving target problem which can cause prototypes to surrender at fluctuating context boundaries. The other 126 neurons represent meaningful sequences. Plot 1 is for reasons of almost perfect symmetry with specialization to symbol **1** reduced to the 63 active neurons that represent the current symbol **0**. These neurons refer to positions in the context space located on the lower horizontal line (⊙). The depicted columns of ⊙,♦ refer to the longest $\{\mathbf{0}, \mathbf{1}\}$-sequences which can be uniquely tracked back in time from the respective neurons. Thus, from bottom to top, the stacked symbols point into the further past of the neurons' temporal receptive fields. Furthermore, it can be observed that the emerged contexts fill the input space in the interval $[0; 1]$ with an almost equidistant spacing. In good agreement with the theory given in the Appendix A, the learned sequences are arranged in a Cantor-like way in the space covered by the neurons' context specializations.

*Representation capability*

The average length of the binary words corresponding to Figure 1 is 5.36; this is quite a good result in comparison to the optimum context representation of random strings with a temporal scope of $\log_2(128) = 7$ time steps per neuron. More structure can be obtained from binary automata exhibiting a bias as a result of choosing asymmetric transition probabilities. We conduct an experiment which has been proposed by Voegtlin in the article [26]. The transition probabilities for a two-state automaton with symbols **0** and **1** are chosen as $P(\mathbf{0}|\mathbf{1}) = 0.4$ and $P(\mathbf{1}|\mathbf{0}) = 0.3$. The other probabilities are then fixed to $P(\mathbf{0}|\mathbf{0}) = 1 - P(\mathbf{1}|\mathbf{0}) = 0.7$ and $P(\mathbf{1}|\mathbf{1}) = 0.6$. The temporally stationary solution for the frequency of **0** is given by the Master equation

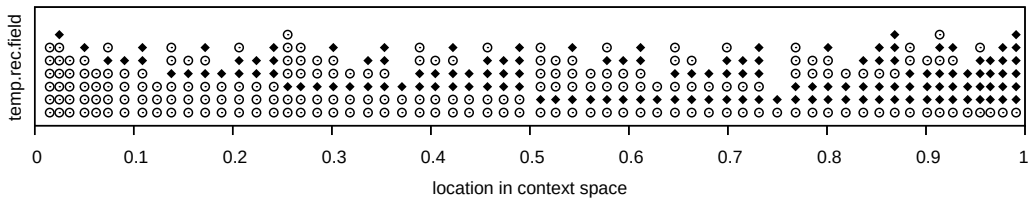


Fig. 1. MNG context associated with the current symbol **0** of a binary sequence. The temporal receptive fields are specialized to sequences of ⊙ for **0** and ♦ for **1**.

$\partial P(\mathbf{0})/\partial t = P(\mathbf{1}) \cdot P(\mathbf{0}|\mathbf{1}) - P(\mathbf{0}) \cdot P(\mathbf{1}|\mathbf{0}) = 0$ for $P(\mathbf{0}) = 4/7$, inducing $P(\mathbf{1}) = 3/7$.

Figure 2 shows neuron histograms that illustrate the effect of the entropy-controlled context influence parameter $\alpha$ for 64 neurons trained on a biased automaton. The first training phase yields two specialized neurons shown in the inset reflecting the symbol probabilities $P(\mathbf{0}) = 4/7$ and $P(\mathbf{1}) = 3/7$. After training, a value of $\alpha = 0.28$ is obtained, for which the neuron activations are displayed by the large histogram in the figure. As a matter of fact, the entropy-based control scheme is suitable for involving all available neurons by allowing the dynamic to account for the unfolding of neurons in the context space.

Another training run has been carried out for the same automaton for 100 neurons with $\gamma = 0.025$, $\beta = 0.45$, and the initial value of $\alpha = 0$ was steered to a final value of 0.37 after $10^6$ pattern presentations; this experimental setup resembles much the one given by Voegtlin [26].

Figure 3 shows, in tree form, the resulting 100 MNG neurons' receptive fields which correspond to the longest words for which neurons are still unique winners. According to the higher frequency of symbol **0** that belongs to left-hand branches, the tree has got a bias towards long sequences for the left. The average word length for this biased automaton is 6.16. The context representation tree is compared to the overall 100 most likely sequences produced by that automaton. As shown by the leaf branches in the figure, many neurons have developed disjoint receptive fields, with the exception of transient neurons, indicated by bullets on the interior tree nodes, for which the descendants and leaves represent still longer sequences. After all, a total number of 63 longest words can be discriminated by MNG, and they are in good correspondence to the most frequent 100 sequences generated by the automaton.

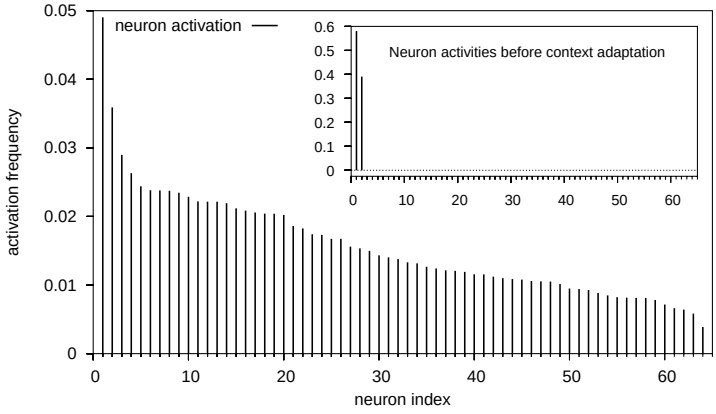


Fig. 2. Sorted neuron activation frequencies for the $\alpha$-entropy control strategy for a biased binary automaton. Inset: before context consideration, two neurons specialize, reflecting the sequence element frequencies $P(\mathbf{0}) = 3/7$ and $P(\mathbf{1}) = 4/7$. Large plot: after entropy maximization, all 64 neurons are involved.

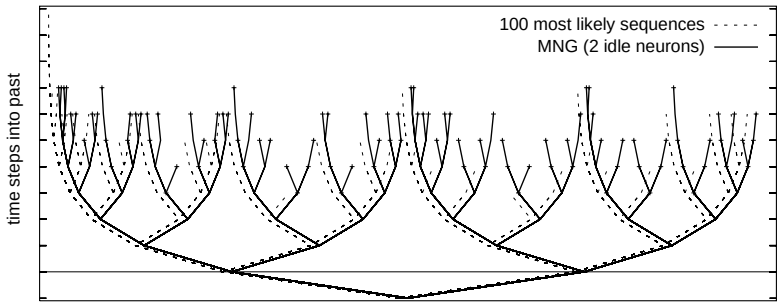


Fig. 3. Tree representation of MNG temporal receptive fields for the biased binary automaton. From bottom to top further past is addressed; the horizontal line marks the (**0**/**1**)-dichotomy of the present instant. Left branching denotes **0**-context, right branching **1**. For comparison, the 100 most likely words of the automaton are displayed in dashed style.

### *4.2 Reber grammar*

So far, the automata experiments deal with sequence elements that can be easily represented as one-dimensional real values. Now, the context representation capabilities of MNG are studied for strings over a 7-letter alphabet. These sequences are generated by the automaton depicted in Figure 4 related to the Reber grammar [15]. The seven symbols have been encoded with pairwise unit distance in a 6-dimensional Euclidean space, analogous to the four corners of a tetrahedron in 3-dimensional space. The concatenation of randomly generated Reber words led to sequences with $3 \cdot 10^6$ input vectors for training and $10^6$ vectors for testing.

For training, 617 MNG neurons are taken for the Reber grammar, a size that can be compared with the HSOM-S architecture [21]. The MNG merge parameter is $\beta = 0.5$, the context influence is initialized to $\alpha = 0$, the starting neighborhood size is $\sigma = 617$, and the context vector is initialized to $\mathbf{0} \in \mathbb{R}^6$ which is the center of gravity of the embedded symbols. The learning rate is $\gamma = 0.03$; after training, the adjusted parameters are $\sigma = 0.5$ and $\alpha = 0.43$.

The context information stored by the ensemble of neurons has been analyzed in two ways: $\langle 1 \rangle$ externally, by using the test sequence to observe characteristic neuron activation cascades, $\langle 2 \rangle$ internally, by backtracking typical activation

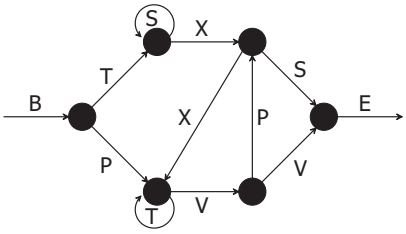


Fig. 4. Reber grammar state graph.

orders of each neuron based on the best matching predecessor context. The latter results rely exclusively on the trained map and not on the training data.

For the externally driven context characterization, the average length of Reber strings from the test sequence leading to unambiguous winner selection is 8.902. Of the 617 available neurons, 428 neurons develop a specialization on distinct Reber words. This is a first indication that the transition properties of the Reber grammar have been faithfully learned. The remaining 189 neurons (30%) represent Reber words as well, but they fall into idle states because their representations are superseded by the better specialized competitors.

Further internal analysis, a one-step context backtracking, has been carried out as follows: for each neuron $i$, all other neurons are visited and the merging vector of their context vector and weight vector is computed; these merging vectors are compared to the context expected by neuron $i$ and sorted according to the distances; a symbol is associated to each neuron in this list by means of its weight vector. On average the first 67.9 neurons in this list corresponding to the best matching contexts represent the same symbol, before neurons for a different symbol are encountered. This large number strongly supports the findings of the external statistics for both high context consistency and proper precedence learning.

Therefore, a three-step backtracking has been performed subsequently in order to collect the 3-letter strings associated with each neuron, strings which are composed of symbols represented by the recursively visited best matching predecessors. The occurrence of these backtracked triplets have been counted. Only valid Reber strings were found, of which the frequencies are given in

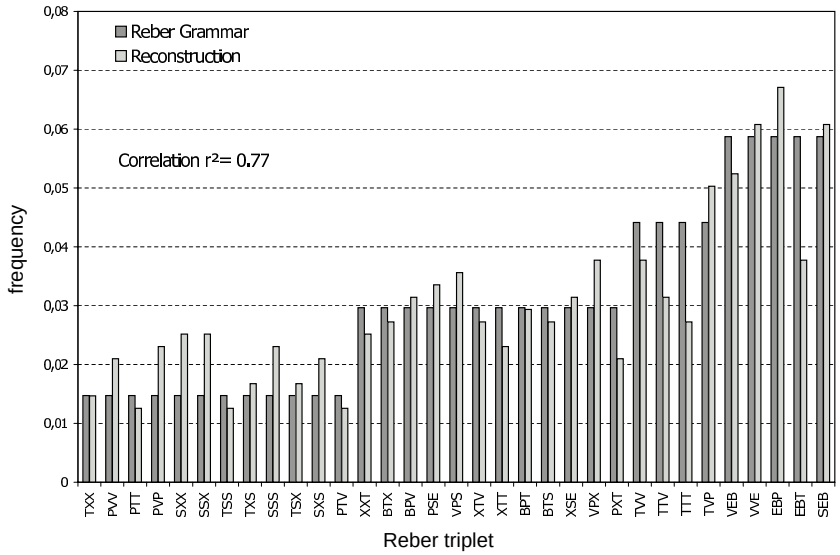


Fig. 5. MSOM frequency reconstruction of trigrams of the Reber grammar. The dark bars show exact Reber triplet frequencies, the light bars show frequencies obtained for MNG. The correlation between both distributions is $r^2 = 0.77$.

Figure 5. In general, the substring frequencies stored in the trained MSOM networks correspond well to the original Reber triplet frequencies, also shown in Figure 5. However, looking closer reveals an underrepresentation of the cyclic **TTT** and the **TTV** transitions, whereas **SSS** is over-represented, and the End–Begin–T-state **EBT** switching is only weakly expressed; a systematical bias cannot be derived easily. Still, the overall correlation of the probabilities of Reber trigrams and the reconstructed frequencies from the map is $r^2 = 0.77$ which is another indication of an adequate neural representation of the Reber transitions.

In a final step, the backtracking and the string assembly has not stopped until the first revisitation of the neuron that started the recursion. This way, words with an average length of 13.78 are produced, most of them validated by the Reber grammar. The longest word

**TVPXTTVVEBTSXXTVPSEBPVPXTVVEBPVVEB**

corresponds almost perfectly to the test data-driven specialization

**TVPXTTVVEBTSXXTVPSEBPVPXTVVE**

that has been determined by keeping track of the winners for the training set. A similarly high correspondence of the shorter strings could be observed for the other neurons. These promising results underline that for a small number of discrete input states a distinctive and consistent representation emerges in the context space.

### *4.3 DNA sequences*

The following real-life example provides clustering of DNA sequences. Thereby, labeling information that is excluded from training will be used to perform a posterior evaluation of the representation capability of the trained network. In other words, classification will be obtained by labeling the neurons subsequent to training, and the classification accuracy on test data will be a rough measure for the generalization ability of the MNG model. DNA data is taken from Sonnenburg who has used and prepared the genome from the *Caenorhabditis elegans* nematode for studies on the recognition of splice sites [18]. These sites mark transitions from expressed to unexpressed gene regions and vice versa, and they are thus important for the assembly of structural and regulatory proteins that constitute and control biochemical metabolisms. Splice site recognition is therefore a topic of interest for the understanding of genotype-phenotype relationships.

Specifically, five pairs of data sets for training and testing of acceptor site detection have been used. Each pair contains a number of 2×10,000 strings

over the $\{\mathsf{C}, \mathsf{G}, \mathsf{T}, \mathsf{A}\}$-alphabet with 50 nucleotides centered around potential canonic $(\mathsf{A}, \mathsf{G})$ splicing boundaries. These strings are labeled by 'true splice site' and 'decoy' (no splice site); after training, these labels will be used for neuron assignments.

For training, the four nucleotides **A**, **G**, **T**, and **C** are encoded as tetrahedron corner points centered around the origin of the 3-dimensional Euclidean space. Furthermore, the DNA strings have been split into a pre-splicing substring of the non-coding intron region and a post-splicing coding exon string to enable the semantic differentiation of both cases. The pre-string has been reversed to first serve, during training, the important nucleotides close to the potential splice location. Independent DNA samples are separated in the model by activating a special neuron that represents the literal null context with undetermined zero weight vector, important for reseting the initial context descriptor of the next DNA string. In each run, a number of 25 cycles has been trained, corresponding to $6 \cdot 10^6$ iterations of 10,000 strings with 24 nucleotides each. Ten runs are calculated with identical training parameters on the five training sets separated into their pre- and post-substrings parts. The MNG networks have 512 neurons and the parameters are learning rate of $\gamma = 0.075$ and an initial context consideration value of $\alpha = 0.001$. The mixing factor for the context descriptor has been set to $\beta = 0.75$ in order to rate the context diversification higher than the specialization to one of only four root symbols.

After training, the resulting context values $\alpha$ adjusted by entropy-based control are located in the large interval $\alpha \in (0.3; 0.6)$. This observation is explained by the fact that, in the final training phase, high specialization with small neighborhoods makes the winner selection insensitive to the weight matching, because weights almost fit perfectly for one of the four symbols; what only matters is a context contribution sufficiently greater than zero. If the splice site detection accuracy is taken for network comparison, the obtained models corresponding to either pre-splice or post-splice DNA strings are mutually very similar in their respective case.

Classification is carried out after unsupervised training in an easy manner: training sequences are presented element by element; thereby the winner neurons count, in associated histograms, the two cases of seeing a sequence marked as splice site or as decoy string. This soft histogram-based labeling is normalized by scaling the bins with the corresponding inverse class frequencies and by scaling with the inverse neuron activation frequency. Then, a test string is presented and a majority vote over the histograms for the winners of the presented symbols determines the class membership. The classes obtained by this voting for all strings in the test set are finally compared against the correct class labels known for the test set.

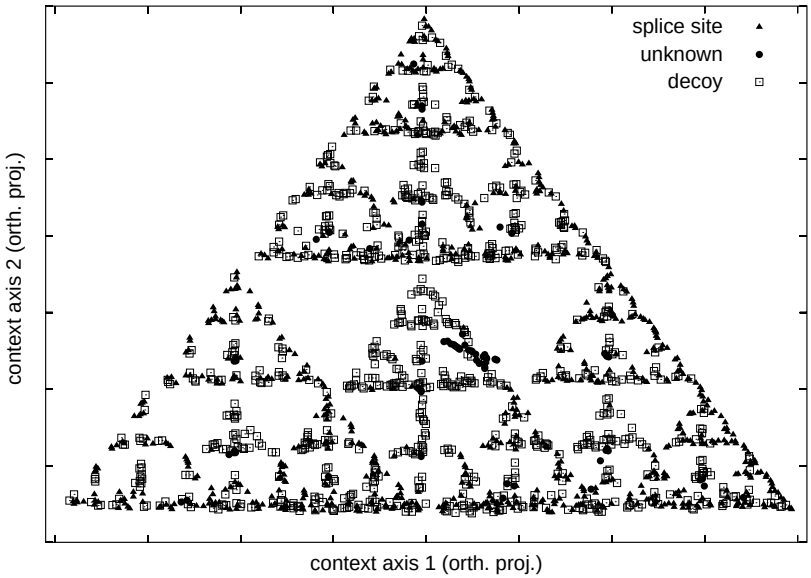


Fig. 6. Prototypes (2048) in MNG context space for DNA intron data with $\beta = 0.5$. The orthogonal 2D-projection of the trained context tetrahedron shows the utilization of the space spanned by the four $\mathbf{C}, \mathbf{G}, \mathbf{T}, \mathbf{A}$-nucleotide vectors. Also, local context specialization can be observed in dependence of splice site and non-splice site strings.

The default guessing classification error is 38.0%, because the overall class frequencies are $P(\text{splice site}) = 0.38$ and $P(\text{decoy}) = 0.62$. For classification based on the post-splice site information, only this baseline value is reached: the average training error is 38.71%±0.60%, the average testing error is 39.43%±0.56%. In contrast to that, for the pre-splice site strings, classification errors of 14.06%±0.66% and 14.26%±0.39% have been obtained for the training and test sets, respectively. Although for supervised classifiers like SVM with the sophisticated TOP kernel the error drops to only 2.2%±0.2%, the result for unsupervised MNG training is still remarkable: the used architecture only takes 512 neurons with 6 dimensions, which is much smaller than the TOP-SVM with ≈1,000 support vectors and ≈5,700 dimensions [18].

The observed differences between the classification results for the pre-splicing intron strings and the post-splicing exon strings are in good correspondence with the biological knowledge about the splicing process: if a $(\mathbf{A}, \mathbf{G})$ pair denotes a true splicing boundary, then it is typically preceded by a pyrimidine-rich region described by approximately 15 nucleotides [14]. MNG is able to capture these consensus strings typical of true splice sites. For the exon strings no such prominent regularity is known, they just re-initiate the coding of the nucleotide to amino-acid transcription without indicating that there has recently been a splice site. These post-splicing nucleotides thus cannot be used for splice site identification by MNG.

Finally, the focus has been put on the context development of the 3D-codes.

Figure 6 displays the 3-dimensional context, obtained by the training of 2,048 neurons on pre-splice site strings with a descriptor mixing parameter of $\beta = 0.5$. For visualization, the context tetrahedron has been rotated to a front view and orthogonally projected to two dimensions: the vertical center line of neurons corresponds to the backside edge. A fractal structure resembling the Sierpinsky gasket becomes clearly visible. At the same time, the neuron labels indicate that this symmetric mixing parameter of $\beta = 0.5$ cannot be used to obtain a clear separation of the splicing versus decoy states. Figure 7 shows the difference for a projection pursuit snapshot of the $2 \times 3$-D neurons maximizing the linear discrimination (LD) of the 2D-projection; 512 neurons are plotted, but now for a mixing parameter of $\beta = 0.75$. Projection pursuit linearly projects the data from high-dimensional space into a low-dimensional visualization space, thereby optimizing the projection matrix parameters according to given separation conditions like LD; for details and references see [2].

For biologically plausible dynamic, it would be desirable to process nucleotide triplets as a whole, because they atomically represent a certain amino acid or a control codon. Related experiments with the required $4^3 = 64$-dimensional encoding are in preparation.

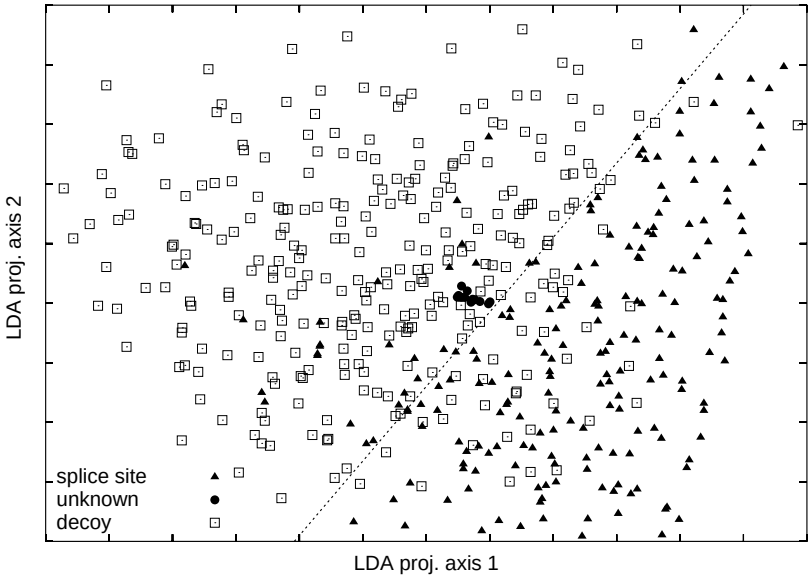


Fig. 7. Prototypes (512) in projected $\mathsf{LD}_{\max}$-space for DNA intron data with $\beta = 0.75$. The 2D-projection of the $2{\times}3$D internal states of the neurons is almost linearly separable into splice site vs. non-splice site; due to the linear $\mathsf{LD}_{\max}$-projection, good separability and clusterization also applies to the original weight/context space.

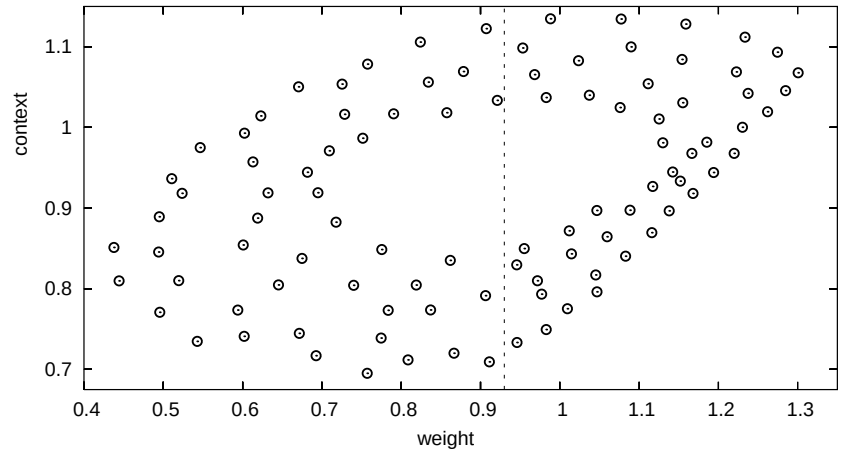


Fig. 8. Prototypes in the MNG context vs. weight space for the Mackey-Glass series. Neuron context has spread well from an initial average value into a large subinterval of the time series domain. A good distribution in the representation space is shown.

### *4.4 Mackey-Glass time series*

An introductory example for processing continuous sequences is the Mackey-Glass time series described by the differential equation $\frac{\mathrm{d}x}{\mathrm{d}\tau} = bx(\tau) + \frac{ax(\tau-d)}{1+x(\tau-d)^{10}}$, using $a = 0.2$, $b = -0.1, d = 17$; a plot of this well-known benchmark series and a histogram is given in Figure 9. The MNG training parameters are a context-preferring mixing value $\beta = 0.75$ and the entropy-controlled $\alpha$ for balancing weight and context in the distance computations; its initialization at $\alpha = 0$ is driven to a final value of $\alpha = 0.18$ after training, which will be detailed below. As shown in Figure 8, the trained MNG context for 100 neurons covers the subinterval $[0.7; 1.15]$ of the input domain $[0.4; 1.3]$. All neurons are well spread in the context vs. weight plane; the vertical line at a weight value of 0.93 marks the average of $10^6$ Mackey-Glass values which serves as common initialization value for the prototype context.

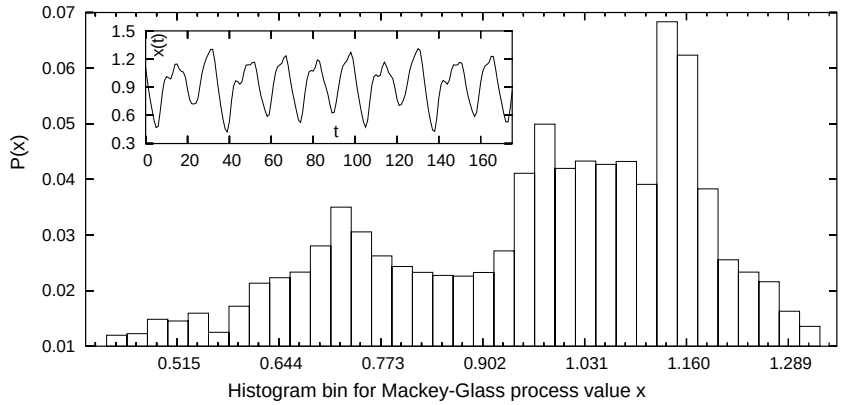


Fig. 9. Histogram for the Mackey-Glass series. The inset shows a part of the series.

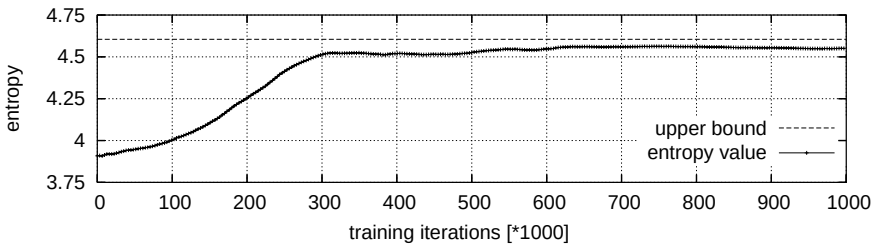


Fig. 10. MNG entropy during Mackey-Glass training. Iteration 0 refers to the point where, after a first context-free specialization phase, the balance parameter $\alpha$ has been released. Entropy saturation, close to the theoretical upper bound of 4.605, corresponds to a final value of $\alpha = 0.18$; see text.

In the first training phase the context influence has been ignored and the prototype weight distribution is — by a magnification functional — closely related to the data distribution shown in the histogram Plot 9. By virtue of the subsequently activated context dynamic, an additional dimension is utilized in order to represent also the history of the currently processed element. Thereby, the bimodal density structure of the Mackey-Glass series is beneficial for the MNG context model to emerge and to unfold in the context space spanned by the data modes. As reported in the article [20], the resulting specialization leads to very small quantization errors for the historic context components.

Here, the focus is put on the neuron activation entropy during training, which shall be maximized by adapting the context metric influence $\alpha$ to an appropriate value. Figure 10 shows the entropy starting from highly specialized neurons without context: small numbers of active neurons yield skewed activity histograms which, as a consequence, correspond to small entropy values. By gradually augmenting the context influence $\alpha$, a richer state space for the distance computation is compelled. This triggers a more detailed winner selection, and the entropy of neuron activations grows. A theoretical upper bound is $H = -\sum_{i=1}^{100} p_i \cdot \log(p_i) = -100 \cdot 1/100 \cdot \log(1/100) = 4.605$ with equal activation probabilities of $p_i = 1/100$ for all neurons. The control strategy for $\alpha$ is a dynamical adaptation at run time, aiming at entropy maximization: $\alpha$ is increased in case of an entropy decay trend and it is decreased otherwise. As shown in Figure 10, entropy saturation takes place at about 300 training cycles with $\alpha$ released after prior context-free weight specialization; however, different learning tasks yield different entropy curves, depending on the relationship of the number of neurons and the number of data modes. After the first weight specialization, the entropy maximization by smooth $\alpha$-adaptation is generally conducive to the context development. Here, spurious entropy fluctuations and instabilities during training are damped by means of a momentum term for $\alpha$. In all experiments presented here, a sampling of about 1,000 entropy values is taken for a training task.

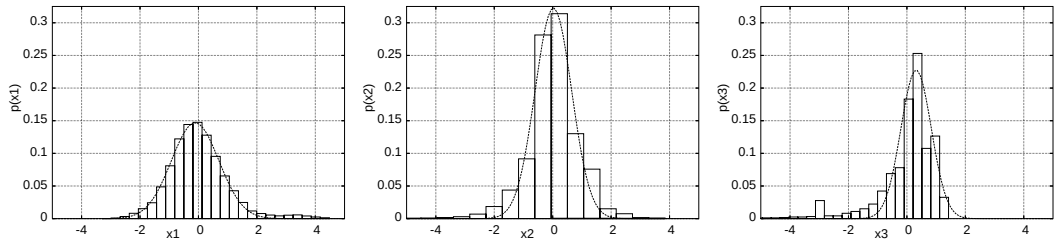


Fig. 11. Histograms of preprocessed B-1991 data columns.

### *4.5 Physiological 3-variate series (B-1991)*

A task that illustrates the limitations of the MNG context for roughly space-filling data has been taken from physionet.org for recordings of a patient suffering from apnea during sleep. The first column contains the heart rate, the second the chest volume, and the third is the blood oxygen concentration. Linear trends, caused by parameter drifts in the gage, have been removed from the data by calculating the first derivative of a smooth order 4 spline interpolation. Mean subtraction has been carried out, and a logarithmic transform $\tilde{x} = \mathrm{sgn}(x) \cdot \log(|x| + 1)$ has been computed for each column to account for the sharp peaks in the otherwise moderate physiological dynamic. Data preprocessing has been concluded by a z-score transform. The final set contains 64,701 samples of temporally dependent 3D-vectors. Although the distributions of each single attribute exhibit a small number of modes, in phase space, the compound temporal 3D data vectors form a curve of concatenated spatial loops around the origin $\mathbf{0}$ with almost unimodal space filling density distribution.

For the obtained series, training has been conducted for 20 cycles. It is assumed that the singular wrap-around for events at the end of the sequence to its beginning do not significantly disturb the training. Figure 12 shows the quantization errors of the training data for three methods, each of them pos-

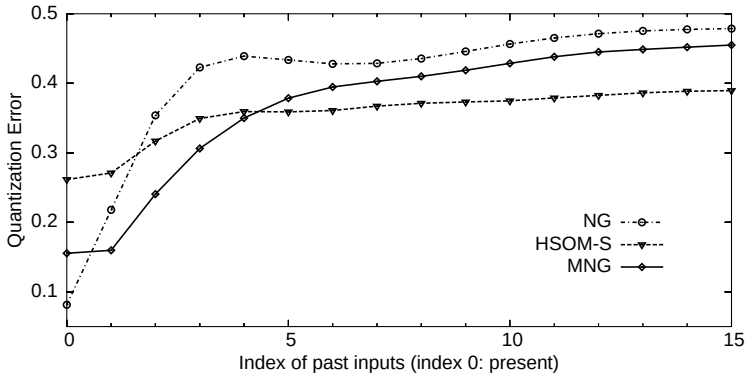


Fig. 12. Temporal quantization errors for the preprocessed B-1991 data. Left: present instant at $t = 0$, right: 15 steps in the past. An error increase over time is apparent for all three models. MNG is best for the recent past $1 < t \leq 4$.

sessing a number of 617 neurons: original neural gas (NG), hyperbolic SOM for structured data (HSOM-S), and MNG. The error refers to the quality of representation for sequence elements located $t$ steps back in time, i.e. the temporal quantization error as defined in [9]. It is expressed by the average standard deviation of the given sequence and the winner unit's receptive field, both considered at the temporal position $t$; thereby, a neuron's receptive field is the average sequence that has occurred just before the winner selection. As expected, NG without context is the worst temporal quantizer, MNG works pretty well up to four steps in the past, but it is then outperformed by HSOM-S.

A particular difficulty of the data set for MNG is caused by the fact that long-term context integration falls into the indistinguishable center of gravity of the data trajectory. Therefore, distinct context clusters cannot easily emerge for histories longer than four steps, and MNG asymptotically results in the standard NG quantization performance. HSOM-S does not suffer from this problem, and it remains on a smaller error level; this is because the HSOM-S context representation does not implement a folding of historic information into the weight domain, but it uses the independent space of explicit back-references to the index of the last winner neuron instead.

A main characteristic of the `B-1991` data set is a noisy trajectory which enters and revisits regions of the phase space from different directions after only a few time steps. Since already many neuron weights are required to represent the densely filled phase space, no storage capacity is left for capturing the temporal context appropriately. However, higher quantization accuracy can be obtained by MNG at higher computational costs, if only a sufficient number of neurons is provided.

### *4.6 Speaker identification by hindsight MNG labeling*

The final experiment processes real-valued speaker data from the UCI repository [1]. In addition to unsupervised clustering, the soft labeling scheme introduced for the DNA strings is used again to obtain a classification model. Data are given as recordings of the Japanese vowel 'ae' from 9 speakers, constituting sequences of 12-dimensional vectors with frequency information. Each utterance is described by a stream of a length between 7 and 29 of temporally connected vectors. In the training set, 30 articulations are available for each speaker, adding up to 270 patterns; in the test set, a variable number of articulations per speaker yields a total of 370 utterances. Each articulation has its own temporal structure; since between different utterances there is no temporal connection, as in the DNA experiment, a special neuron is added to

[1] `http://kdd.ics.uci.edu/databases/JapaneseVowels/JapaneseVowels.html`

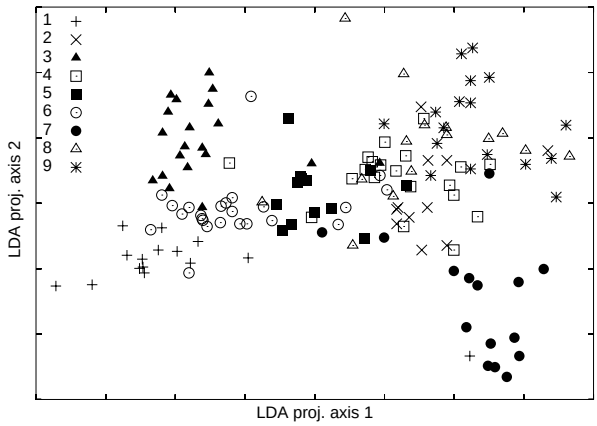


Fig. 13. LDA-projection of the weight × context-space of speaker data.

represent the default state $\boldsymbol{w} = \boldsymbol{c} = E(\mathbf{X})$ when no context is available at an utterance start.

Training has been realized for 150 neurons without taking into account the speaker identity. The initial neuron context is set to the center of the training data, the context descriptor's mixing is a fair value of $\beta = 0.5$, and a large learning rate of $\gamma = 0.3$ could be used. The initial context influence parameter is set to $\alpha = 0.01$ and driven to a final value of $\alpha = 0.37$ during entropy maximization.

Figure 13 supports a good clusterability of the (weight × context)–product space $\mathbb{R}^{12} \times \mathbb{R}^{12}$: the according 24-dimensional prototypes have been projected to two dimensions by means of the linear discriminant analysis (LDA) technique [6]; the parameters of the 24×2-projection matrix have been adapted by projection pursuit to yield a maximum Gaussian separation of the prototypes in the two-dimensional target space [5]. Hence in the unprojected original space, a good separability can be expected, which is confirmed by the experiment.

Analogous to the DNA experiment, after training, each neuron is assigned a 9-bin histogram containing activation frequencies for the speakers from the training set. For each articulation sequence in the test set, the accumulated majority vote over the bins of activated neurons is calculated to identify the speaker. In contrast to the DNA splice site detection task, the resulting histograms are very specific for the speakers. When, finally, the obtained posterior labels are checked against the data, there is no error on the training set and an error of only 2.7% on the test set. This accuracy is much better than the reference errors of 5.9% and 3.8% accompanying the original data set. For MNG training with 1,000 neurons, a number of 21 neurons fall into idle state during context formation, and the error decreases to only 1.6%. This interesting result illustrates that good-natured interpolation emerges from the dynamic, if more neurons than training examples are given.

## 5 Conclusions

We have investigated the MSOM merge context model for temporally or spatially connected sequential data. Self-organizing context is obtained by a back-reference to the winner in the previous time step, compactly described by a weighted linear combination of the last winner's weight and context. MSOM context can be combined with other self-organizing architectures such as NG or LVQ. For unsupervised sequence processing with accurate temporal quantization, MSOM has been integrated into NG in this work.

In the first part of this article, theoretical properties of the model have been discussed. In the limit of vanishing neighborhood cooperation, the proposed recursive context definition leads to an efficient fractal encoding as the fixed point of the training dynamic. This fact brings MSOM into line with TKM and RSOM which are also based on fractal encoding of sequences in weight space. In contrast to that, SOMSD and RecSOM extend the context representations to the separate space of topological previous winner indices. However, unlike TKM and RSOM, MSOM models context explicitly and this gained adaptation flexibility allows MSOM to simulate finite automata.

¿From the supplied experiments it becomes clear that sophisticated data preprocessing is not mandatory in order to obtain good temporal data models with MSOM; for discrete symbols just a canonic real-value vector encoding should be provided, and for initialization convenience, data should be zero-centered. As a consequence of self-organization, the capacity of the context representation grows with the number of neurons. After initial specialization without context contribution, a maximum utilization of neurons is obtained by maximizing the network entropy via the context influence parameter $\alpha$. Since recurrent neural networks cover the supervised learning tasks pretty well, most applications of the merge context model are found for unlabeled data. Nevertheless, the DNA splice site identification and especially the speaker recognition task, both realized by means of soft *a posteriori* neuron labeling, indicate that we can trust in the found context representations and that there is some potential for processing labeled data too.

Further work should consider metrics more general than the squared Euclidean distance. Also systematic studies on the interplay of the context influence and the context update strength are of interest. Another even more challenging question, if there is a way to generalize the presented robust MSOM model from sequences to more complex graph structures, remains to be answered in future.

**Acknowledgements**

Many thanks to the anonymous reviewers for helping to improve the quality of this manuscript by their valuable comments.

## A Proof of Theorem 1

The properties of the merge context is studied in two steps: ⟨1⟩ the optimum choices for the prototype weight and the context are investigated; ⟨2⟩ it is proved that these optima result from the training dynamic as stable fixed points. We assume that neighborhood cooperation is neglected, i.e. only the winner is adapted according to the given learning rule. For the winner, this assumption applies to both MSOM and MNG.

⟨1⟩ The best choice of neuron $i$ is the one for which weight $\boldsymbol{w}^i$ and context $\boldsymbol{c}^i$ yield $\mathrm{d}_i(\mathbf{x}^t) = 0$ for the current pattern $\mathbf{x}^t$:

$$\begin{aligned}\mathrm{d}_i(\mathbf{x}^t) &= (1-\alpha)\cdot\|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + \alpha\cdot\|\mathsf{c}^t - \boldsymbol{c}^i\|^2 \\ &= (1-\alpha)\cdot\|\mathbf{x}^t - \boldsymbol{w}^i\|^2 + \alpha\cdot\|(1-\beta)\cdot\boldsymbol{w}^{I_{t-1}} + \beta\cdot\boldsymbol{c}^{I_{t-1}} - \boldsymbol{c}^i\|^2 \,.\end{aligned}$$

$I_{t-1}$ is the index of the winner in the previous time step. Both squared summands can be considered separately. The left one trivially becomes the minimum of 0, if the weight vector represents the input pattern, i.e. for $\boldsymbol{w}^i = \boldsymbol{w}^{opt(t)} = \mathbf{x}^t$. Then, by induction, the right one expands to

$$\begin{aligned}\boldsymbol{c}^i = \boldsymbol{c}^{opt(t)} &= (1-\beta)\cdot\mathbf{x}^{t-1} + \beta\cdot\boldsymbol{c}^{I_{t-1}} \\ &= (1-\beta)\cdot\mathbf{x}^{t-1} + \beta\cdot\boldsymbol{c}^{opt(t-1)} \\ &= (1-\beta)\cdot\mathbf{x}^{t-1} + \beta\cdot\big((1-\beta)\cdot\mathbf{x}^{t-2} + \beta\cdot(\ldots + \beta\cdot(1-\beta)\cdot\mathbf{x}^1)\ldots\big) \\ &= \sum_{j=1}^{t-1}(1-\beta)\cdot\beta^{j-1}\cdot\mathbf{x}^{t-j} \quad \text{(note: } j-1 \text{ is exponent, } t-j \text{ is index)}\end{aligned}$$

with the assumption of zero context at the sequence start $\mathbf{x}^1$.

⟨2⟩ Now, focusing on the convergence of a neuron that is specialized on a particular sequence element within its unique context, asymptotically stable fixed points of the training update dynamic are obtained. Still, we neglect neighborhood cooperation, and we assume that enough neurons are available for generating disjoint optimum weight settings for each presented pattern.

The analysis of iterative weight adaptation in presence of the target vector yields:

$$\|\boldsymbol{w}^{I_t} + \gamma \cdot (\mathbf{x}^t - \boldsymbol{w}^{I_t}) - \mathbf{x}^t\| = (1-\gamma) \cdot \|\boldsymbol{w}^{I_t} - \mathbf{x}^t\| \quad \Rightarrow \quad \boldsymbol{w}^{I_t} \to \mathbf{x}^t \,.$$

This describes an exponential convergence because of $\gamma \in (0;1)$. Analogously,

$$\|\boldsymbol{c}^{I_t} + \gamma \cdot \left((1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}} - \boldsymbol{c}^{I_t}\right) - \boldsymbol{c}^{opt(t)}\| \;\Rightarrow\; \boldsymbol{c}^{I_t} \to \boldsymbol{c}^{opt(t)}$$

describes the context convergence, if

$$(1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}} \to \boldsymbol{c}^{opt(t)}$$

can be shown. With $\boldsymbol{w}^{I_{t-1}} \to \mathbf{x}^{t-1}$ and by induction of $\boldsymbol{c}^{I_{t-1}} \to \boldsymbol{c}^{opt(t-1)}$ with $\boldsymbol{c}^{I_1} \to \mathsf{c}^1 = 0 = \boldsymbol{c}^{opt(1)}$:

$$\begin{aligned}(1-\beta) \cdot \mathbf{x}^{t-1} + \beta \cdot \boldsymbol{c}^{opt(t-1)} &= (1-\beta) \cdot \mathbf{x}^{t-1} + \beta \cdot \sum_{j=2}^{t-1} (1-\beta) \cdot \beta^{j-2} \cdot \mathbf{x}^{t-j} \\ &= \sum_{j=1}^{t-1} (1-\beta) \cdot \beta^{j-1} \mathbf{x}^{t-j} = \boldsymbol{c}^{opt(t)} \quad \square\end{aligned}$$

This sum for $\boldsymbol{c}^{opt}$ denotes a fractal encoding of the context vector in the weight space, which is known to be a very compact and efficient representation [23].

## B Proof of Theorem 2

Assume that a finite automaton $A = (\Sigma, S, \delta)$ is given. We construct a network with merge context which simulates $A$. We assume that distances are computed using the $L_1$-norm $\|\mathbf{x}^1 - \mathbf{x}^2\|_1 = \sum_i |x_i^1 - x_i^2|$ and the parameters are set to $\alpha = \beta = 0.5$. We refer to the weight and context of a neuron by the symbols $\boldsymbol{w}$ and $\boldsymbol{c}$. The recursive network dynamic for an input sequence $s = (\mathbf{x}^1, \ldots, \mathbf{x}^t)$ is given by the equation

$$\mathrm{d}_j(\mathbf{x}^t) = 0.5 \cdot \|\mathbf{x}^t - \boldsymbol{w}^j\|_1 + 0.5 \cdot \|\mathsf{c}^t - \boldsymbol{c}^j\|_1 \,,$$

where $\mathsf{c}^1$ is a fixed vector and

$$\mathsf{c}^t = 0.5 \cdot \boldsymbol{w}^{I_{t-1}} + 0.5 \cdot \boldsymbol{c}^{I_{t-1}} \,.$$

$I_{t-1}$ is the index of the winner in the previous time step.

We now show that the transition function of the automaton can be implemented by this dynamics. The transition function $\delta$ can be interpreted as a

boolean function based on the primitives which test whether the current symbol is a specific symbol $\sigma_i$ and the current state is a specific state $\mathrm{sta}_j$. We can state this formula in the following form: assume $\mathtt{I}$ is a variable for the current input state and $\mathtt{S}$ a variable for the current input symbol. $\mathtt{O}$ is a variable the output state. Then we can formulate the function as a disjunction of implications

$$((\mathtt{I} = \mathrm{sta}_{i_1} \wedge \mathtt{S} = \sigma_{j_1}) \vee \ldots \vee (\mathtt{I} = \mathrm{sta}_{i_l} \wedge \mathtt{S} = \sigma_{j_l})) \rightarrow (\mathtt{O} = \mathrm{sta}_k))$$

corresponding to all possible output states of $\delta$. Thus, to implement $\delta$, we have to test a disjunction of conjunctions for the precondition of the respective implication. This will be done in two steps, one for the computation of the disjunction, one for the conjunction. However, within a MSOM, no variables, states, and symbols are explicitly available and we need to interpret appropriate activation patterns of MSOM as a distribution of truth values among the variables of $\delta$. For this purpose, we identify the neurons with an $|S| \times |\Sigma|$ array, and we identify the rows (i.e. all neurons in a row are 'on') with input symbols and the columns with states. This is the main idea of the construction. For the concrete implementation, some minor technical issues are to be added, such that the activation of rows, columns, and the computation of the above boolean formulas can be implemented by a recursive iteration of distance computations in MSOM.

Weights and contexts are elements of $\mathbb{R}^d = \mathbb{R}^{|S|\cdot|\Sigma|+4}$. We refer to the components of a vector $\mathbf{x}$ in this space by tuples $x_{ij}$ for the first $|S| \cdot |\Sigma|$ entries, $1 \leq i \leq |S|$ and $1 \leq j \leq |\Sigma|$, and $x_i$ for the last 4 entries, $1 \leq i \leq 4$.

First, we define the neurons of the map and their weight and context vectors. The idea behind this construction is as follows: the last four positions act as a counter; they ensure that only a neuron of a specified group can become winner. For the other positions, we identify the state $\mathrm{sta}_i$ with the vector $\mathbf{x} \in \mathbb{R}^d$ with $x_{ij} = 1$ for all $j$, and all remaining entries are 0; we identify $\sigma_j$ with the vector $\mathbf{x}$ with $x_{ij} = 1$ for all $i$, all remaining entries are 0. We construct neurons which simulate the transition function $\delta$ of the automaton in two steps. The first step identifies input tuples $(\mathrm{sta}_i, \sigma_j)$ of $\delta$ to ensure that the internal context $\mathtt{c}$ of the map is 1 for position $\mathtt{c}_{ij}$ and 0 for any other position. The second step collects, for a fixed state $\mathrm{sta}_k$, all tuples $(\mathrm{sta}_i, \sigma_j)$ which are in the inverse image of $\mathrm{sta}_k$ under $\delta$. The output of this step is an internal context which corresponds to the state $\mathrm{sta}_k$. These two steps are both divided into two sub-steps, one which actually performs the computation and another one which is responsible for a 'cleaning' of the result.

For every pair we define $(\mathrm{sta}_i, \sigma_j) \in S \times \Sigma$ two neurons of the map,

- $n(\text{sta}_i, \sigma_j)$ with weight vector

$$w_{kl} = \begin{cases} 1 & \text{if } l = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad w_k = \begin{cases} 2 \cdot d & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases}$$

  and context vector

$$c_{kl} = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad c_k = 0 \quad \text{for all } k\,.$$

- $n^{\texttt{clean}}(\text{sta}_i, \sigma_j)$ with weight vector

$$w_{kl} = 0 \quad \text{for all } k \text{ and } l \quad \text{and} \quad w_k = \begin{cases} 2 \cdot d & \text{if } k = 2 \\ 0 & \text{otherwise} \end{cases}$$

  and context vector

$$c_{kl} = \begin{cases} 1 & \text{if } k = i \text{ and } l = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad c_k = 0 \quad \text{for all } k\,.$$

For every state $\text{sta}_i \in S$ we choose two neurons of the map

- $n(\text{sta}_i)$ with weight vector

$$w_{kl} = \begin{cases} 4 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad w_k = \begin{cases} 2 \cdot d & \text{if } k = 3 \\ 0 & \text{otherwise} \end{cases}$$

  and context vector

$$c_{kj} = \begin{cases} 0.5 & \text{if } \delta(\text{sta}_k, \sigma_j) = s_i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad c_k = \begin{cases} \texttt{GAP}_i & \text{if } k = 3 \\ 0 & \text{otherwise} \end{cases},$$

  whereby $\texttt{GAP}_i = 0.5 \cdot \Big(\texttt{MAX} - \Big|\{(\text{sta}_l, \sigma_j) \in S \times \Sigma \mid \delta(\text{sta}_l, \sigma_j) = \text{sta}_i\}\Big|\Big)$. $\texttt{MAX}$ is defined as $\max_k \Big|\{(\text{sta}_l, \sigma_j) \in S \times \Sigma \mid \delta(\text{sta}_l, \sigma_j) = \text{sta}_k\}\Big|$
- $n^{\texttt{clean}}(\text{sta}_i)$ with weight vector

$$w_{kl} = 0 \quad \text{for all } k \text{ and } l \quad \text{and} \quad w_k = \begin{cases} 2 \cdot d & \text{if } k = 4 \\ 0 & \text{otherwise} \end{cases}$$

and context vector

$$c_{kl} = \begin{cases} 2 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad c_k = 0 \quad \text{for all } k\,.$$

The encoding function enc of an element $\sigma_j \in \Sigma$ is given by the vector $\mathbf{x}^j$ with

$$x^j_{kl} = \begin{cases} 1 & \text{if } l = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad x^j_k = \begin{cases} 2 \cdot d & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases},$$

which is followed by three input vectors $\mathbf{0}^i$ with $i = 2, 3, 4$ which entries equal

$$0^i_{kl} = 0 \text{ for all } k \text{ and } l \quad \text{and} \quad 0^i_k = \begin{cases} 2 \cdot d & \text{if } k = i \\ 0 & \text{otherwise} \end{cases}.$$

Thus, $\text{enc}(\sigma_j) = (\mathbf{x}^j, \mathbf{0}^2, \mathbf{0}^3, \mathbf{0}^4)$. Finally, we identify a context $\mathsf{c}(\text{sta}_i)$ for every state $\text{sta}_i \in S$, setting

$$\mathsf{c}(\text{sta}_i)_{kj} = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \mathsf{c}(\text{sta}_i)_k = \begin{cases} d & \text{if } k = 4 \\ 0 & \text{otherwise} \end{cases}.$$

The initial context $\mathsf{c}^1$ corresponds to the state $\text{sta}_1$: $\mathsf{c}^1 := \mathsf{c}(s_1)$.

Now we show the following: assume $\delta(\text{sta}_i, \sigma_j) = s_k$; then, starting from $\mathsf{c}(\text{sta}_i)$, the input $\mathbf{x}^j$ followed by three entries $\mathbf{0}^2$, $\mathbf{0}^3$, and $\mathbf{0}^4$ yields the winner neuron $n^{\mathtt{clean}}(\text{sta}_k)$ and the internal context $\mathsf{c}(\text{sta}_k)$. In particular — by applying this result recursively — $A$ accepts a sequence $s$ iff the winner for $\text{enc}(s)$ is neuron $n^{\mathtt{clean}}(\text{sta}_{|S|})$.

Assume the context is $\mathsf{c}(\text{sta}_i)$, the input is $\mathbf{x}^j$, followed by $\mathbf{0}^2$, $\mathbf{0}^3$, and $\mathbf{0}^4$. The last four components of the inputs make sure that the winner neuron for input $\mathbf{x}^j$ is of the form $n(\text{sta}_l, \sigma_m)$, the winner for $\mathbf{0}^2$ is of the form $n^{\mathtt{clean}}(\text{sta}_l, \sigma_m)$, the winner for $\mathbf{0}^3$ is of the form $n(\text{sta}_l)$, and the winner for $\mathbf{0}^4$ is of the form $n^{\mathtt{clean}}(\text{sta}_l)$. The other places simulate the function $\delta$. For convenience, an example of the simulation of a computation step $\delta$ is provided in Table B.1.

In general, we obtain the following distances and internal contexts in four consecutive recursive steps:

Step 1: context $\mathsf{c}(\text{sta}_i)$, input $\mathbf{x}^j$:

· Neuron $n(\mathrm{sta}_i, \sigma_j)$ yields the distance $0.5 \cdot d$ and the context vector $\mathtt{c}$ with

$$\mathtt{c}_{ml} = \begin{cases} 1 & \text{if } m = i \text{ and } l = j \\ 0.5 & \text{if } m = i \text{ and } l \neq j \\ 0.5 & \text{if } m \neq i \text{ and } l = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \mathtt{c}_m = \begin{cases} d & \text{if } m = 1 \\ 0 & \text{otherwise} \end{cases}$$

· Neuron $n(\mathrm{sta}_i, \sigma_m)$ with $m \neq j$ yields the distance $0.5 \cdot d + |S|$.

| input | | | | context | | | | winner weight | | | | winner context | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | 1 | | 1 | | 1 | | | | 1 | | 1 | |
| 1 | | | | | | | | 1 | | | | | | | |
| 1 | | | | | | | | 1 | | | | | | | |
| 20 | | | | | | | 10 | 20 | | | | | | | |
| | | | | 1 | | 0.5 | | | | | | 1 | | | |
| | | | | 0.5 | | | | | | | | | | | |
| | | | | 0.5 | | | | | | | | | | | |
| | 20 | | | 10 | | | | | 20 | | | | | | |
| | | | | 0.5 | | | | | | | | 0.5 | | | |
| | | | | | | | | 4 | | 4 | | 0.5 | | 0.5 | |
| | | | | | | | | | | | | | | | |
| | | 20 | | | 10 | | | | | 20 | | | | 0.5 | |
| | | | | 0.25 | | | | | | | | | | | |
| | | | | 2.25 | | 2.25 | | | | | | 2 | | 2 | |
| | | | | | | | | | | | | | | | |
| | | | 20 | | | 10 | | | | | 20 | | | | |

Table B.1
An example of the computation steps for $|S| = 3$ and $|\Sigma| = 2$, i.e. $d = 6 + 4$. In every block of the table, the input, the internal context, and the weight and context vector of the winner are depicted. Thereby, the components of the vectors are denoted using the convention introduced above, i.e. using $|S| = 3$ rows with $|\Sigma| = 2$ columns, and an additional row with 4 entries. The input for $\delta$ is $\sigma_1$ and the context is $\mathrm{sta}_1$. Assume $\delta(\mathrm{sta}_1, \sigma_1) = \mathrm{sta}_2$. In addition, assume that the pairs $(\mathrm{sta}_2, \sigma_1)$ and $(\mathrm{sta}_2, \sigma_2)$ yield $\mathrm{sta}_2$. Assume $\mathtt{MAX} = 3$. All entries which are not shown yield 0.

- Neuron $n(\text{sta}_m, \sigma_j)$ with $m \neq i$ yields the distance $0.5 \cdot d + |\Sigma|$.
- Neuron $n(\text{sta}_m, \sigma_l)$ with $m \neq i$ and $l \neq j$ yields the distance $0.5 \cdot d + |S| + |\Sigma| - 1$.
- Neurons $n^{\texttt{clean}}(\text{sta}_m, \sigma_l)$, $n(\text{sta}_m)$, and $n^{\texttt{clean}}(\text{sta}_m)$ yield a distance of at least $2 \cdot d$.

Step 2: the context $\texttt{c}$ of Step 1 and input $\mathbf{0}^2$:

- Neuron $n^{\texttt{clean}}(\text{sta}_i, \sigma_j)$ yields a distance $0.5 \cdot d + 0.25 \cdot (|S| - 1) + 0.25 \cdot (|\Sigma| - 1)$. The resulting context vector $\texttt{c}$ is of the form

$$\texttt{c}_{ml} = \begin{cases} 0.5 & \text{if } m = i \text{ and } l = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \texttt{c}_m = \begin{cases} d & \text{if } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

- Neuron $n^{\texttt{clean}}(\text{sta}_i, \sigma_m)$ with $m \neq j$ and neuron $n^{\texttt{clean}}(\text{sta}_m, \sigma_j)$ with $m \neq i$ yield a distance of $0.5 \cdot d + 0.25 \cdot (|S| - 1) + 0.25 \cdot (|\Sigma| - 1) + 0.5$.
- Neuron $n^{\texttt{clean}}(\text{sta}_m, \sigma_l)$ with $m \neq i$ and $l \neq j$ has got the distance $0.5 \cdot d + 0.25 \cdot |S| + 0.25 \cdot |\Sigma| + 0.5$.
- Neurons $n(\text{sta}_m, \sigma_l)$, $n(\text{sta}_m)$, and $n^{\texttt{clean}}(\text{sta}_m)$ yield distances of at least $2 \cdot d$.

Step 3: the context $\texttt{c}$ of Step 2 and input $\mathbf{0}^3$ (remember that $\delta(\text{sta}_i, \sigma_j) = \text{sta}_k$):

- Neuron $n(\text{sta}_k)$ yields a distance $2 \cdot |\Sigma| + 0.5 \cdot d + 0.25 \cdot \texttt{MAX} - 0.25$ and the context vector $\texttt{c}$ with

$$\texttt{c}_{ml} = \boldsymbol{\xi} + \begin{cases} 2 & \text{if } m = k \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \texttt{c}_m = \begin{cases} d + 0.5 \cdot \texttt{GAP}_i & \text{if } m = 3 \\ 0 & \text{otherwise} \end{cases}$$

  whereby $\boldsymbol{\xi} \in \mathbb{R}^{|S| \times |\Sigma|}$ is a vector with entries 0.25 for indices $(m, l)$ with $\delta(\text{sta}_m, \sigma_l) = \text{sta}_k$ and 0 for any other entry.
- Neuron $n(\text{sta}_m)$ with $m \neq k$ yields the distance $2 \cdot |\Sigma| + 0.5 \cdot d + 0.25 \cdot \texttt{MAX} + 0.25$.
- Neurons $n(\text{sta}_m, \sigma_l)$, $n^{\texttt{clean}}(\text{sta}_m, \sigma_l)$, and $n^{\texttt{clean}}(\text{sta}_m)$ yield distances of at least $2 \cdot |\Sigma| - 0.5 + 2 \cdot d$.

Step 4: the context $\texttt{c}$ of Step 3 and input $\mathbf{0}^4$:

- Neuron $n^{\texttt{clean}}(\text{sta}_k)$ yields distance $0.5 \cdot d + 0.25 \cdot \texttt{GAP}_i + 0.125 \cdot (\texttt{MAX} - 2\texttt{GAP}_i) = 0.5 \cdot d + 0.125 \cdot \texttt{MAX}$. The context vector $\texttt{c}$ with

$$\texttt{c}_{ml} = \begin{cases} 1 & \text{if } m = k \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \texttt{c}_m = \begin{cases} d & \text{if } m = 4 \\ 0 & \text{otherwise} \end{cases}$$

  arises. This is $\texttt{c}(\text{sta}_k)$.
- Neuron $n^{\texttt{clean}}(\text{sta}_m)$ for $m \neq k$ yields a distance of at least $0.5 \cdot d + 0.25 \cdot \texttt{GAP}_i + 2 \cdot |S| - 0.125 \cdot |S| + 0.125 \cdot (\texttt{MAX} - 2 \cdot \texttt{GAP}_i) = 0.5 \cdot d + 0.125 \cdot \texttt{MAX} + 1.75 \cdot |S|$.
- For all other neurons, a distance of at least $2 \cdot d$ arises. Note that $\texttt{MAX} < d$.

This concludes the proof.

## C Proof of Theorem 3

Given a finite automaton $A = (\Sigma = \{\sigma\}, S, \delta)$ and a delay $\texttt{delay}$. Assume $\text{enc} : \Sigma \to \mathbb{R}^d$ is an encoding function and a $\mathsf{TKM}$ is given, such that the $\mathsf{TKM}$ simulates $A$ with this delay. Input sequences for $A$ just differ with respect to their length, since the alphabet is unary. Assume $\text{enc}(\sigma) = (\mathbf{x}^1, \ldots, \mathbf{x}^{\texttt{delay}})$. Then an input sequence $s \in \Sigma^*$ of length $t$ yields the activation

$$\sum_{j=1}^{t} \alpha \cdot (1-\alpha)^{\texttt{delay}\cdot(t-j)} \cdot \Big( (1-\alpha)^{\texttt{delay}-1} \cdot \|\mathbf{x}^1 - \boldsymbol{w}^i\|^2 + \ldots + (1-\alpha) \cdot \|\mathbf{x}^{\texttt{delay}-1} - \boldsymbol{w}^i\|^2 + \|\mathbf{x}^{\texttt{delay}} - \boldsymbol{w}^i\|^2 \Big)$$

of neuron $i$ with weight vector $\boldsymbol{w}^i$. The term $W_i := \alpha \cdot \sum_{j=1}^{\texttt{delay}} (1-\alpha)^{\texttt{delay}-j} \cdot \|\mathbf{x}^j - \boldsymbol{w}^i\|^2$ is independent of $t$, thus the activation can be rewritten as

$$W_i \cdot \sum_{j=1}^{t} (1-\alpha)^{\texttt{delay}\cdot(t-j)} \,.$$

Because the term $\sum_{j=1}^{t} (1-\alpha)^{\texttt{delay}\cdot(t-j)}$ is identical and positive for all neurons $i$, the winner depends only on the quantity $W_i$. In particular, the winner is the same for every input sequence. Thus, $A$ is the trivial automaton.

## References


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

[2] D. Cook, A. Buja, J. Cabrara, and C. Hurley. Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistics*, 4:155–172, 1995.

[3] M. Cottrell, J. Fort, and G. Pagés. Two or three things that we know about the Kohonen algorithm. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 235–244. D-side Publications, 1994.

[4] I. Farkas and R. Miikulainen. Modeling the self-organization of directional selectivity in the primary visual cortex. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 251–256. Springer, 1999.

[5] J. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[6] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[7] M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, 2003.

[8] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35, 2004.

[9] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, to appear, 2004.

[10] B. Hammer and T. Villmann. Mathematical aspects of neural networks. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 59–72. D-side Publications, 2003.

[11] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.

[12] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Temporal sequence processing using recurrent som. In *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Engineering Systems*, volume 1, pages 290–297, Adelaide, Australia, 1998.

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

[14] R. Padgett, P. Grabowski, M. Konarska, S. Seiler, and P. Sharp. Splicing of messenger rna precursors. *Annual Review of Biochemistry*, 55:1119–1150, 1986.

[15] A. Reber. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6:855–863, 1967.

[16] H. Ritter. Self-organizing maps on non-euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–110. Elsevier, Amsterdam, 1999.

[17] J. Sinkkonen and S. Kaski. Clustering based on conditional distribution in an auxiliary space. *Neural Computation*, 14:217–239, 2002.

[18] S. Sonnenburg. New Methods for Splice Site Recognition. Diplom thesis, Institut für Informatik, Humboldt-Universität zu Berlin, 2002.

[19] M. Strickert, T. Bojer, and B. Hammer. Generalized relevance LVQ for time series. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 677–683. Springer, 2001.

[20] 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.

[21] M. Strickert and B. Hammer. Unsupervised recursive sequence processing. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 433–439. D-side Publications, 2003.

[22] M. Strickert and B. Hammer. Self-organizing context learning. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 39–44. D-side Publications, 2004.

[23] P. Tino and G. Dorffner. Predicting the future of discrete sequences from fractal representations of the past. *Machine Learning*, 45(2):187–217, 2001.

[24] M. Varsta, J. del R. Milan, and J. Heikkonen. A recurrent self-organizing map for temporal sequence processing. In *Proc. ICANN'97, 7th International Conference on Artificial Neural Networks*, volume 1327 of *Lecture Notes in Computer Science*, pages 421–426. Springer, Berlin, 1997.

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

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