Look and Feel What and How Recurrent Self-Organizing Maps Learn

Jérémy Fix, Hervé Frezza-Buet

Look and feel what and how recurrent self-organizing maps learn

Jérémy Fix and Hervé Frezza-Buet

LORIA, CNRS, CentraleSupélec, Université Paris-Saclay, F-57000 Metz, France,
{jeremy.fix, herve.frezza-buet}@centralesupelec.fr

Abstract. This paper introduces representations and measurements for revealing the inner self-organization that occurs in a 1D recurrent self-organizing map. Experiments show the incredible richness and robustness of an extremely simple architecture when it extracts hidden states of the HMM that feeds it with ambiguous and noisy inputs.

Keywords: recurrent self-organizing map, sequence processing, hidden markov models

1 Introduction

Self-organizing maps (SOMs) or Kohonen maps, introduced in [9], is a particular topographically organized vector quantization algorithm. It computes a mapping from a high dimensional space to a usually one or two dimensional regular grid with the specificity that close positions in the regular grid are associated with close positions in the original high dimensional space. We have a pretty good understanding of what a SOM is doing. Even if there is no energy function associated with the Kohonen learning rule which could formally state what Kohonen maps do actually capture (some authors actually suggested some alternative formulations derived from an energy function, see for example [6]), we can still pretty much see Kohonen maps as a K-means with a topology i.e. capturing the distribution of input samples in a topographically organized fashion. As soon as we experiment with, for example, 2D-Kohonen maps with two dimensional input samples, we quickly face the nice unfolding of the map sometimes trapped in some kind of local minima where there remains some twist in the map. While our understanding of Kohonen SOMs is pretty clear, the things become more complicated when we turn to recurrent SOMs.

Recurrent SOMs are a natural extension of SOMs when dealing with serial inputs in order to "find structure in time" (J. Elman). This extension follows the same principle introduced for supervised multi-layer perceptrons by [7,3] of feeding back a context computed from the previous time step. These recurrent SOMs are built by extending the prototype vector with an extra component which encapsulates some information about the past. There are indeed various proposals about the information memorized from the past, e.g. keeping only the location of the previous best matching unit [4] or the matching over the whole

map[11]. An overview of recurrent SOMs is provided in [5]. Cellular and biologically inspired architectures have been proposed as well [8]. When the question of understanding how recurrent SOMs work comes to the front, there are some theoretical results that bring some answers. However, as any theoretical study, they are necessarily limited in the questions they can address. For example, [10] studied the behavior of recurrent SOMs by analyzing its dynamics in the absence of inputs. As for usual SOMs for which mathematical investigations do not cover the whole field yet [2], these theoretical results bring only a partial answer and there is still room for experimental investigation. Despite numerous works, how the recurrent SOMs deal with serial inputs and what they actually learn is not obvious: "The internal model representation of structures is unclear" [5]. We indeed lack the clear representations that we possess for understanding SOMs.

In order to tackle this issue, we focus in this paper on the simplest recurrent SOM where the temporal context is only the position of the best matching unit (BMU) within the map at the previous iteration (which bears resemblance to the SOM-SD of [4]). This simplicity comes with the ability to design specific visualizations to investigate the behavior of the map. As we shall see in the experiments, despite this simplicity, there is still an interesting richness of dynamics. In particular, we will investigate and visualize the behavior of this simple recurrent SOM when inputs are provided sequentially by different hidden Markov models. These will illustrate the behavior of the recurrent SOM in the presence of ambiguous observations, long-term dependencies, changing dynamics, noise in the observations and noise in the transitions.

2 Methods

2.1 Algorithm

Let us consider a stream of inputs $\xi^t \in X$ available at each successive time step t. Here, let us use $X = [0, 1]$. Let us consider a topological set $\mathcal{M}$ where unit (sometimes called neuron) positions lie. We use a 1D map in this paper, thus $\mathcal{M} = [0, 1]$ is considered with a topology induced by the Euclidean distance. The map is made of N units, each unit is denoted by an index $i \in \mathcal{I} \subset \mathcal{M}$. Indexes are equally spread over $\mathcal{M}$, i.e. $\mathcal{I} = \{0, 1/(N-1), 2/(N-1) \cdots, 1\}$.

For the sake of introducing notations for our recurrent algorithm, let us start by rephrasing the self-organizing map algorithm (SOM). Each unit i is equipped with an input weight $w_i \in X$ also referred to as a *prototype*. When ξ^t is presented to the map, all the units compute a matching value $\mu_i = \mu(w_i - \xi^t)$, where $\mu(d) = \max(1 - |d|/\rho_\mu, 0)$ is a linear decreasing function that reaches 0 for the distance value ρ_μ[1]. The best matching unit (BMU) $i_\star$ can be computed here from μ_i as $i_\star = \text{argmax}_i \, \mu_i$. It could have been computed directly as the unit for which $|w_i - \xi^t|$ is minimal, as usual SOM formulation do, but the current formulation involving μ_i allows for the forthcoming extension to recurrent SOM. Once $i_\star$ is determined consecutively to the submission of ξ^t, the prototypes in

[1] A more classical Gaussian function could have be used as well.

the neighborhood of $i_\star$ have to be updated. The strength of that update for any w_i is $\alpha \mathrm{h}\left(i - i_\star\right)$, with $\alpha \in [0,1]$ and $\mathrm{h}\left(d\right) = \max\left(1 - |d|/\rho_\mathrm{h}, 0\right)$. Let us stress here that the width ρ_h of the learning kernel is kept constant, as opposed to usual SOM and recursive SOM implementations where it continuously decreases.

The recurrence is added to our formulation of SOM by using *context* weights $c_i \in \mathcal{M}$. They are trained in the same way as w_i, except that they are fed with $i_\star^{t-1}$ instead of ξ^t. A context matching distribution $\mu'_i = \mu\left(c_i - i_\star^{t-1}\right)$ is computed as we did for μ_i. The BMU needs to be determined from both matchings. To do so, each unit computes a global matching $\mu''_i = \frac{\mu_i + \mu'_i}{2}$, such as the BMU is determined as $i_\star = \mathrm{argmax}_i\, \overline{\mu''}_i$. The overline of $\overline{\mu''}$ indicates a low pass spatial filtering with a gaussian kernel of standard deviation σ_k. Moreover, the selection of the BMU is done by randomly sampling in the set of possible BMUs. These two elements improve the algorithm in our settings where observations are drawn from a discrete set. The whole process studied in this paper can then be formalized into algorithm 1. For all the experiments, we used $N = 500$ units, a neighbour kernel width $\rho_\mathrm{h} = 0.05$, a learning rate $\alpha = 0.1$, a matching sensitivity $\rho_\mu = 0.4$ and a Gaussian convolution kernel standard deviation $\sigma_k = 0.0125$.

Algorithm 1 Architecture update at time t.

1: Get ξ^t, compute $\forall i \in \mathcal{I},\ \mu_i = \mu\left(w_i - \xi^t\right),\ \mu'_i = \mu\left(c_i - i_\star^{t-1}\right)$
2: $\forall i \in \mathcal{I},\ \mu''_i = (\mu_i + \mu'_i)/2$
3: $i_\star^t \in \mathrm{argmax}_i\, \overline{\mu''}_i$ // $\overline{\mu''} = \mu'' * k$, $i_\star$ is taken randomly in argmax.
4: $\forall i \in \mathcal{I}$
$$\begin{cases} w_i^t = w_i^{t-1} + \alpha \mathrm{h}\left(i - i_\star\right) . \left(w_i^{t-1} - \xi^t\right) \\ c_i^t = c_i^{t-1} + \alpha \mathrm{h}\left(i - i_\star\right) . \left(c_i^{t-1} - i_\star^{t-1}\right) \end{cases}$$

In our experiments, the inputs in X that are provided at each time step are generated from a Hidden Markov Model (HMM). The HMM has a finite set $S = \{s_0, s_1, \cdots\}$ of states. Each state is an integer (i.e. $S \subset \mathbb{N}$). At each time step, a state transition is performed according to a transition matrix. In the current state s^t, the observation is sampled from the conditional probability $\mathbb{P}\left(\xi \mid s^t\right)$, defined by the observation matrix of the HMM. Different states of the HMM may provide a similar observation. In this case, the recursive architecture is expected to make the difference between such states in spite of the observation ambiguity. In other words, the current BMU $i_\star^t$ value is expected to represent the actual s^t even if several other states could have provided the current input ξ^t.

2.2 Representations

Algorithm 1 can be executed with any dimension for $\mathcal{M}$ without loss of generality. Nevertheless, we use 1D maps ($\mathcal{M} = [0,1]$) for the sake of visualization. Weights w_i are in $X = [0,1]$ as well. They can be represented as a gray scaled value, from black (0) to white (1). In the bottom left part of figure 1, the background of the chart is made of w_i^t, with t in abscissa and i in ordinate. On this

chart, red curves are also plotted. This is done when the HMM is deterministic (and thus cycling through its states, visiting $s_0, s_1, \cdots, s_{p-1}, s_0, s_1, \cdots$). If the state sequence that is repeated throughout the experiment has a length p ($p = 10$ in experiment of figure 1), p red curves are plotted on the chart. For $0 \leq k < p$, the kth red curve links the points $\{(t, i^t_\star) \mid t \bmod p = k\}$. The curves show the evolution of the BMU position corresponding to each of the p states throughout learning. From left to right in that chart in figure 1, some red curves are initially overlaid before getting progressively distant. Such red curves splits show a bifurcation since the map allocates a new place on $\mathcal{M}$ for representing a newly detected HMM state. This allocation has a topography since the evolution is a split and then a progressive spatial differentiation of the state positions.

Let us take another benefit from using 1D maps and introduce an original representation of both w and c weights. This representation is referred to as a *protograph* in this paper. It consists of a circular chart (see three of them on top left in figure 1). The gray almost-closed circle represents $\mathcal{M} = [0, 1]$. At time step t, one can plot on the circle the two weights related to $i^t_\star$. First weight, related to the input, is $w\left(i^t_\star\right)$, which is a value in X to which a gray level is associated. This is plotted as a gray dot with the corresponding gray value, placed on the circle at position $i^t_\star$. The second weight to be represented for $i^t_\star$ is $c\left(i^t_\star\right)$, related to the recurrent context, which is a position in $\mathcal{M}$ and thus a position on the circle. $c\left(i^t_\star\right)$ is represented with an arrow, starting from position $c\left(i^t_\star\right)$ on the circle and pointing at $i^t_\star$ on the circle, where the dot representing $w\left(i^t_\star\right)$ is actually located. This makes a dot-arrow pair for $i^t_\star$. The full *protograph* at time t plots the dot-arrow pairs $\left(w\left(i_\star\right), c\left(i_\star\right)\right)$ for the 50 last steps. The third protograph in figure 1 seems to contain only 10 dot-arrow pairs since many of the 50 ones are identical to others. This last protograph corresponds to an organized map, it reveals the number of states visited by the HMM (number of dots), where they are encoded in the map (dot positions), which observation each state provides (dot colors), and the state sequence driven by the HMM transitions (follow the arrows from one state to another). Making movies from the succession of such protographs unveils the dynamics of the organization of spatio-temporal representations in the map. The splits and separation mentioned for the red curves is then visible as a split of one dot into two dots that slide afterwards away one from the other. Movies of the experiments are available online[2].

2.3 Evaluation

The representations presented so far enables to unveil the inner dynamics of a single run. Nevertheless, the ability of the architecture to encode the hidden states of the HMM providing the inputs needs to be measured quantitatively from several runs (a thousand in our experiments). At time step t, let us store the dataset $D^t = \left\{\left(i^{t-99}_\star, s^{t-99}\right), \cdots, \left(i^{t-1}_\star, s^{t-1}\right), \left(i^t_\star, s^t\right)\right\}$ that is a 100-sized sliding window containing the last observed BMU position / HMM state pairs. If the map encodes the HMM states with a dedicated BMU position, each observed

[2] `http://www.metz.supelec.fr/~fix_jer/recsom1D`

BMU position must be paired with a single state. In this case, D^t can be viewed as a set of samples of a *function* from $\mathcal{M}$ to S. To check this property for the map at time t, a supervised learning process is performed from D^t, that is viewed here as an input/output pairs container. As $S \subset \mathbb{N}$, this is a multi-class learning problem. A basic bi-class decision stump is used in this paper (i.e. a threshold on map position values makes the decision), adapted to the multi-class problem thanks to a one-versus-one scheme. Let us denote by χ^t the classification error rate obtained on D^t (i.e. the empirical risk). The value χ^t is null when one can recover the state of the HMM from the position of the BMUs collected during the 100 steps. It is higher when a small contiguous region of the map is associated with several HMM states.

In our experiment, χ^t is computed every 100 steps in a run. As previously said, 1000 runs are performed in order to compute statistics about the evolution of χ^t as the map gets organized. At each time step t, only the best 90% of the 1000 χ^t are kept. The evolution curve, as reported in the right of figure 1, plots the upper and lower bounds of these 900 values, as well as their average. There are indeed less than 10% of the runs for which the map does not properly self-organize. A deeper investigation of this phenomenon is required, but it is out of the scope of the present paper, which is focused on the dynamic of the self-organization when it occurs. This is why the corresponding runs are removed from the performance computation.

3 Results

As mentioned in section 2.1, the serial inputs ξ^t are observations provided by the successive states of a HMM. Let us use a comprehensive notation for the HMMs used in our experiments. Observations are in $X = [0, 1]$ as previously stated and 6 specific input values are represented by a letter ($\mathtt{A} = 0, \mathtt{B} = 0.2, \mathtt{C} = 0.4, \mathtt{D} = 0.6, \mathtt{E} = 0.8, \mathtt{F} = 1$). The HMM denoted by `AABFCFE` is then a 7-state HMM for which s_0 provides observation `A`, s_1 provides `A` as well, s_2 provides `B`, ... s_6 provides `E`. The states are visited from s_0 to s_6 periodically. In this particular HMM, (s_0, s_1), as well as (s_3, s_5) are ambiguous since they provide the same observation (`A` and `F`) as an input to the recurrent SOM. When a state provides an observation uniformly sampled in $[0, 1]$, it is denoted by $*$. The notation $\mathtt{AB}\overline{\mathtt{CD}}^{\sigma}\mathtt{EF}$ means that values for both s_2 and s_3 are altered by an additive normal noise with standard deviation σ. Last, the notation $\mathtt{ABC}\big|_q^p\mathtt{DEF}$ means that the HMM is made of two periodical HMMs `ABC` and `DEF`, with random transitions from any of the state of `ABC` to any of the state of `DEF` with a probability p. Random transitions from `DEF` to `ABC` occurs similarly with a probability q.

3.1 Ambiguous observations

In order to test the ability of the recurrent SOM to deal with ambiguous observations, we consider the HMM `ABCEFEDCB`, i.e. a HMM with 10 states and 6 observations. There are 8 states which provide an observation that is ambiguous

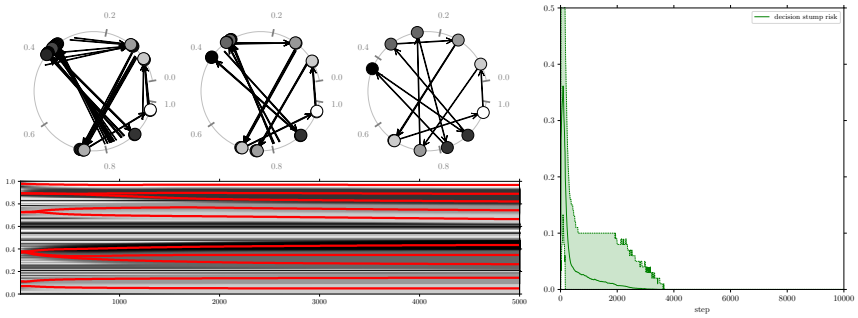

Fig. 1. Observations from `ABCDEFEDCB`. The protographs are recorded at $t = 200, 500, 5000$.

(the state cannot be identified given only the current observation). The recurrent SOM receives observations during 5000 time steps, i.e. 500 presentations of the full sequence. A single run is depicted on the left of figure 1. The SOM initially captures the individual observations, irrespective of their context (at $t = 200$, approximately 6 units are the BMUs), and then, ambiguous observation begin electing their own BMUs and the recurrent prototypes begin to reflect the context of the observations. This splitting of the winning positions ultimately leads to one distinct BMU for each state of the HMM and, at $t = 5000$, the structure of the HMM can be uncovered from the weights $(w(i_\star), c(i_\star))$ as displayed in the respective protograph. Red curves show how the 6 areas in the map insensitive to the context split to build the expected appropriate 10 areas. Running the same experiment for 1000 runs indicates that the ability of the algorithm to identify the structure of the HMM is statistically significant (right of figure 1). The observations produced by this HMM could be easily disambiguated, taking into account the observation of the previous time step. In the next experiment, we study longer term dependencies.

3.2 Long term dependencies

In the second experiment, the algorithm receives observations from the HMM `AAAAAAAF`. This experiment seeks to test if the algorithm can capture long-term dependencies. Indeed, this HMM produces exactly the same observation `A` for a fixed number of steps before outputting `F` which seeds the ambiguity of `A`. The experiment is run for 5000 steps, i.e. 625 repetitions of the full sequence. The results of a single run are displayed on the left of figure 2. Initially, the SOM captures the two observations `A` and `F` independently of the context (see the plot of the prototypes on the top left of the figure). Then, we observe several units specializing to the observation of `A` in a context dependent manner. The logic of the propagation of the context can be appreciated from the split of the red curves. The first state producing a `A` to be clearly identified is the one associated with the unit with the smallest position (the node shown in back on the first

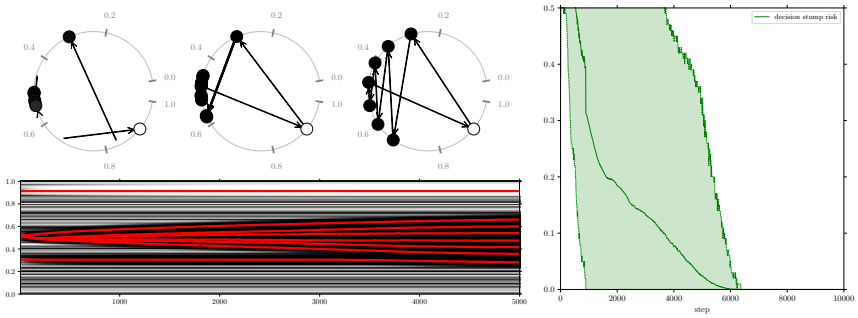

Fig. 2. Observations from `AAAAAAAF`. The protographs are recorded at $t = 40, 1000, 5000$.

protograph around position 0.3). This is the simplest to be identified because it is the state just after the one outputting `F`. Then, the dependence on the context propagates through all the previous steps. The $c\,(i_\star)$ weights are continuous over $\mathcal{M}$, as for usual weights in SOMs. In the specific run of figure 2, when we consider the black dots counterclockwise, the arrow origins $c\,(i_\star)$ progress clockwise, i.e. $c\,(i_\star)$ is a monotonously decreasing function. Running this experiment on 1000 independent trials reveals that the algorithm is able to capture the structure of this HMM (see figure 2, right).

3.3 Adapting to a changing dynamics

In this third experiment, the algorithm receives observations from the HMM `ABCDEFEDCB` for the 10000 first steps and then from the HMM `ABCBAFEDEF` for the last 10000 steps. The prototypes obtained at $t = 10000$ and $t = 20000$ as well as the evolution of the observation weights and winner locations are displayed on the left of figure 3 for a single run. The algorithm successfully recovers the structure of the two HMM. Analyzing the red curves at the time the second HMM is presented is illuminating. One can note there is a reuse of the previously learned prototypes and some adaptation of the prototypes. Indeed, there was a single BMU responsive for a `F` (white node on the first protograph) for the first sequence which splits and two BMUs are now responsive for a `F` for the second sequence, which makes sense given the second HMM has two different states producing the observation `F`. The same comment holds for the BMUs when the observation `A` is produced by the HMM. On the contrary, while two BMUs had observation prototypes w close to a `C` and `D` during the first training period, only one BMU is remaining for each `C` and `D` after learning with the second HMM.

The performance of the algorithm ran for 1000 independent trials is shown on the right of figure 3. Similarly to the first experiment, it takes around 5000 steps to learn the sequence. At the time the HMM is changed, there is a degradation in the performances that quickly drops.

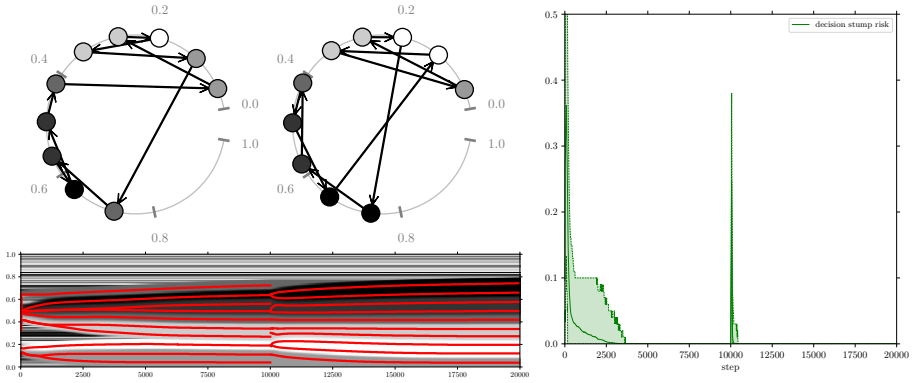

Fig. 3. Observations from ABCDEFEDCB for the first 10000 steps and then from ABCBAFEDEF.The protographs are recorded at $t = 10000, 20000$.

3.4 Noisy observations

We now perform an experiment to test the robustness of the algorithm in the presence of noise in the observations. This experiment involves the HMM $\overline{\texttt{BCDEDC}}^{0.05}$, i.e. each observation is perturbed with a normal noise of standard deviation 0.05. Given that the unperturbed observations are separated by 0.2, a normal noise of standard deviation 0.05 leads to slightly overlapping observations. On the left of figure 4, one can recognize the 6 states identified by the algorithm. The blur in the representation of the prototypes comes from the fact that the circles arranged along the ring are displayed with a transparency proportional to the winning frequency of the displayed BMU. Given the observations are noisy, there is jitter in the location of the BMUs but still, the elected BMUs for each state of the HMM remains in a compact set. The sequence $\texttt{B}, \texttt{C}, \texttt{D}, \texttt{E}, \texttt{D}, \texttt{C}$ tends to elicit BMUs in positions around 0.5, 0.65, 0.1, 0.35, 0.2 and 0.8. Running the experiment on 1000 runs, the performance χ^t decreases almost down to 0.0 as shown on the right of figure 4. This confirms that the locations that are BMUs for a given HMM state are indeed compact sets. Finally, while the absence of noise and the linear neighborhood function kept the BMUs confined within the map in the previous experiments, the presence of noise in this experiment ultimately leads to populate all the map; all the positions within the map tend to be recruited to encode the HMM.

3.5 Perturbed by a noise state

In the last experiment, we consider a challenging HMM $\texttt{ABCDEFEDCB}\big|_q^p *$, with $p = 0.03$ and $q = 0.1$. This HMM is based on the sequence ABCDEFEDCB. There is a probability p for the HMM to jump from any of the states ABCDEFEDCB to the state we denote $*$ which emits a uniformly distributed observation. There is also a probability q to jump from the state $*$ back to one of the states in

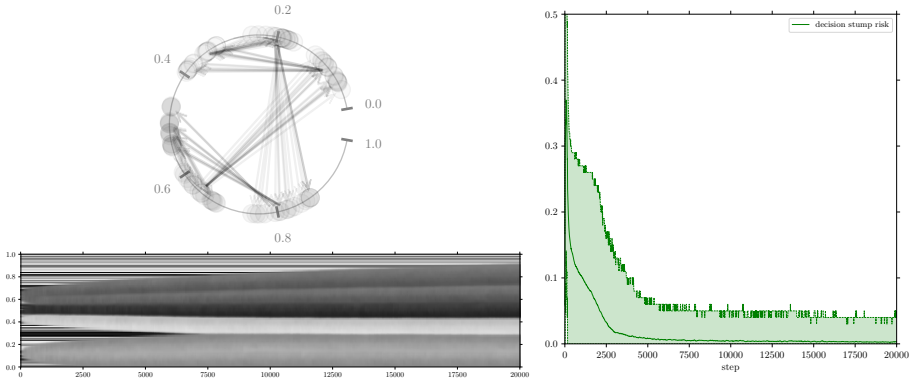

Fig. 4. Observations from $\overline{\texttt{BCDEDC}}^{0.05}$. The protograph is recorded at $t = 20000$.

ABCDEFEDCB. The clean sequence of observations from ABCDEFEDCB is therefore regularly corrupted with a uniform noise. This challenging HMM can mimic for example a temporal disruption of sensors. With $p = 0.03$, there is a probability $(1-p)^9 = 0.76$ to completely unroll the sequence ABCDEFEDCB when starting from A. When the HMM is in the state $*$, it stays in this state for $\frac{1}{q} = 10$ steps in average. The results of a single run are displayed on the left of figure 5. The structure of the HMM, without the noisy state, can be recognized from the plot of the prototypes on the top of the figure (if we omit the BMUs just before $i = 0.2$ and just after $i = 0.8$). It should be noted that the structure of the algorithm does not allow it to capture the noisy state and the latter is therefore filtered by the algorithm. Running the experiment for 1000 runs indicates that the ability of the algorithm to capture the structure of the clean HMM is statistically significant, as shown on the right of figure 5. For computing the statistics on figure 5, the samples labelled with $*$ are removed from the dataset D^t. However, this still does not lead to a perfect classification. Indeed, when the HMM is back from the noisy state $*$, it sometimes requires two successive observations to identify in which state the HMM is. This explains why χ^t is not null.

4 Conclusion

This paper presents an empirical approach of recurrent self-organizing maps by introducing original representations and performance measurements. The experiments show how spatio-temporal structure gets organized internally to retrieve the hidden states of the external process that provides the observations. An area of the map associated with an observation splits into close areas when observation ambiguity is detected, and then areas get progressively separated onto the map. Unveiling the emergence of such a complex and continuous behavior, from both the SOM-like nature of the process and a simple re-entrance, is the main result of this paper. Such a simple architecture also shows robustness to

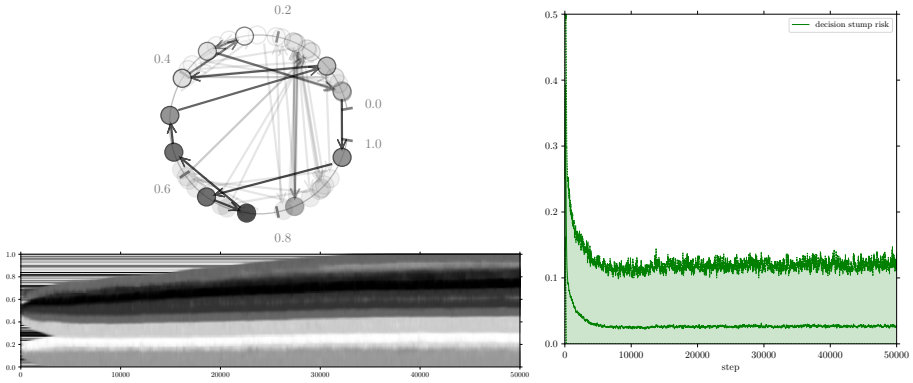

Fig. 5. Observations from $\texttt{ABCDEFEDCB}\Big|_q^p *$. The protograph is recorded at $t = 20000$.

temporal and spatial damages in the input series, as well as the ability to deal with deep time dependencies while the recurrence only propagates previous step context. Forthcoming work will consist in using such recurrent maps in more integrated multi-map architecture, as started in [1].

Acknowledgement: This work is supported by the European Interreg Grande Région / Région Grand-Est project GRONE.

References

1. Baheux, D., Fix, J., Frezza-Buet, H.: Towards an effective multi-map self organizing recurrent neural network. In: ESANN. pp. 201–206 (2014)
2. Cottrell, M., Fort, J., Pags, G.: Theoretical aspects of the SOM algorithm. Neurocomputing 21(1), 119 – 138 (1998)
3. Elman, J.L.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
4. Hagenbuchner, M., Sperduti, R., Tsoi, A.C., Member, S.: A self-organizing map for adaptive processing of structured data. IEEE Trans. Neural Netw. 14, 491–505 (2003)
5. Hammer, B., Micheli, A., Sperduti, A., Strickert, M.: Recursive self-organizing network models. Neural Netw. 17(8-9), 1061 – 1085 (2004)
6. Heskes, T.: Energy functions for self-organizing maps. In: Oja, E., Kaski, S. (eds.) Kohonen Maps, pp. 303 – 315. Elsevier Science B.V. (1999)
7. Jordan, M.I.: Serial order: A parallel distributed processing approach. Tech. Rep. 8604, Institute for Cognitive Science, University of California, San Diego (1996)
8. Khouzam, B., Frezza-Buet, H.: Distributed recurrent self-organization for tracking the state of non-stationary partially observable dynamical systems. Biologically Inspired Cognitive Architectures 3, 87–104 (2013)
9. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. 43(1), 59–69 (1982)
10. Tiňo, P., Farkaš, I., van Mourik, J.: Dynamics and topographic organization of recursive self-organizing maps. Neural Computation 18(10), 2529–2567 (2006)
11. Voegtlin, T.: Recursive self-organizing maps. Neural Netw. 15(8–9), 979–991 (2002)