ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Towards an effective multi-map self organizing recurrent neural network

Denis Baheux, Jérémy Fix and Hervé Frezza-Buet *

Supelec - MaLIS team
UMI 2958 Georgia Tech/CNRS
2 rue Edouard Belin 57070 METZ - France

Abstract. This paper presents a multi-map joint self-organizing architecture able to represent non-markovian temporal sequences. The proposed architecture is inspired by previous works based on dynamic neural fields. It provides a faster and easier to handle architecture making it easier to scale to higher dimensional machine learning problems.

1 Introduction

Self-organization is a feature commonly observed in nature, since natural processes, that are incredibly robust, are far from being driven by a master process that knows how to organize things. Natural processing rather relies on the cooperation of local elements in a population, where a suitable response to environmental constraints emerges. This way of computing is dramatically different from nowadays computational designs and this is why such processing are of primary interest for computer science. In this paper, we focus on self-organizing processes inspired from the cerebral cortex neural tissue that Teuvo Kohonen [1] paved the way for. Indeed, self-organizing maps (SOM) originate from an attempt to model distributed competitive processes over the cortical surface, viewed as a neural field [2], where a localized patch of few selected neurons is the locus of learning processes. Driving self-organization with a neural field has been reported as being difficult [3] since the neural field dynamics is hard to control. Nevertheless, cortically-inspired self-organization is already widespread in machine learning, since the SOM algorithm gets rid of neural fields dynamics by using a winner-take-most (WTM) selection process. Although the computation of an argmax involved in WTM is less biologically plausible than a neural field, it has the great advantage of enabling a fast computation in SOM, while preserving self-organizing properties.

In past studies, we have explored multi-map self-organization, where competition processes in each map were coupled [4]. This has led us to design multi-modal learning but also, more recently, temporal sequence learning through the use of time-delay connections [5]. The core dynamical process of these multi-map approaches relies on recurrent connections between the maps, making the map competitions mutually dependent. Nevertheless, this smart resonance process relies on coupled neural fields and thus requires a lot of computational resources as well as handling complex dynamical processes.

*This study has been supported by the Man Robot Dialog (MaRDi) project sponsored by Agence Nationale de la Recherche

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

This paper presents an attempt to set up a multi-map resonant joint self-organization process with computational techniques that are more suitable for efficient machine learning implementations. As Kohonen did for single map self-organization, the idea is to get rid of neural fields while preserving the dynamical properties of the whole architecture. The proposed mechanisms will be illustrated in the context of temporal sequence processing, in order to see whether the results in [5] can be obtained with the architecture proposed here.

2 A joint self-organizing maps architecture

Inspiring from [4], the proposed architecture can be viewed as a construction set where building blocks are self-organizing maps, connected together. The experiment addressed in this paper, following [5], involves two of them, as shown in Fig. 1, where each map is represented by a group of three scalar distributions. Any map $\mathcal{M}$ is a bi-dimensional arrangement of computational units, so that each unit m can be referred to by its discrete position $p_m \in [0..N]^2$ in a 2D-grid. The positions actually used lie inside a disk included in the grid, in order to avoid side effects induced by the corners, but this does not impact the forthcoming descriptions. Let us note $x_m \in [0,1]^2$ the normalized positions for further convenience. The output computed by some map $\mathcal{M}$ results from the selection of a position over the map surface. This output, denoted by $O_\mathcal{M} \in [0,1]^2$, is thus simply the selected position (normalized).

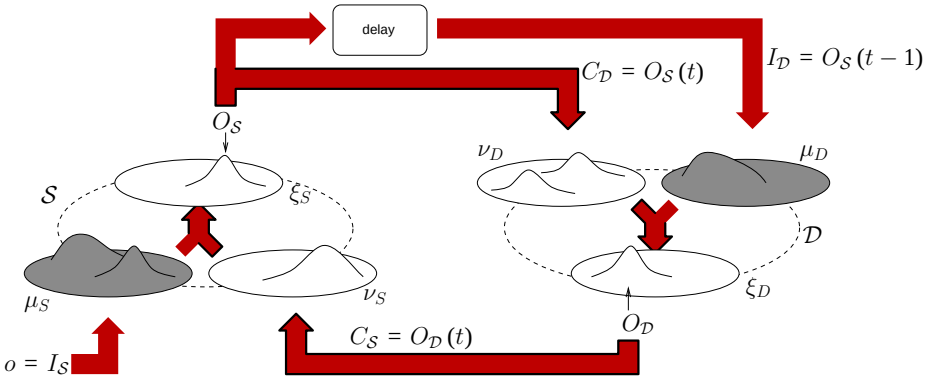


Fig. 1: Multimap resonant self-organization. See text for details.

The map outputs a selected position according to some competition process, based on the matching of some input value $I_\mathcal{M}$ against the unit prototypes, as for classical SOM. Nevertheless, here, the prototypes are twofold. First the *input prototype* of unit m, denoted by w_m, represents the "preferred" input value of unit m. It lives in the same space as the actual map input. For all units m, the value resulting from matching the input against the prototype is defined as

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

$$\forall m,\ \mu_m = \exp\left(-\|I_{\mathcal{M}} - w_m\|^2 / 2{\sigma_\mu}^2\right). \tag{1}$$

The second prototype handled by m is referred to as a *context prototype*, denoted by ω_m. It is matched against the context input $C_{\mathcal{M}}$ which is here defined as the output $O_{\mathcal{N}} \in [0,1]^2$ of some remote map $\mathcal{N}$, ie. a normalized position in that map, so $\omega_m \in [0,1]^2$ as well. The context matching value ν_m is defined by a Gaussian similarly:

$$\forall m,\ \nu_m = \exp\left(-\|C_{\mathcal{M}} - \omega_m\|^2 / 2{\sigma_\nu}^2\right),\ C_{\mathcal{M}} = O_{\mathcal{N}} \tag{2}$$

For each unit m in $\mathcal{M}$, the two matching values μ_m and ν_m are merged into a global matching ξ_m as follows:

$$\forall m,\ \xi_m = \sqrt{\mu_m \left(\beta\mu_m + (1-\beta)\,\nu_m\right)}, \tag{3}$$

with $\beta \in [0,1]$. This merge ensures that the context modulates a matching mainly driven by the input, therefore the network is not creating artificial global matchings ("hallucinations") independent from the inputs.

The prototype update rule is related to the current output value of the map, whose actual computation is explained further. This update is the one used in SOMs and it is here the same for both w_m and ω_m. Let us stress here that, as opposed to the SOM algorithm, all the learning parameters are kept constant over time, enabling the network to preserve its plasticity.

$$\begin{array}{l} \forall m \in \mathcal{M},\ w_m \leftarrow w_m + \alpha h\left(x_m, O_{\mathcal{M}}\right)\left(I_{\mathcal{M}} - w_m\right) \\ \forall m \in \mathcal{M},\ \omega_m \leftarrow \omega_m + \alpha h\left(x_m, O_{\mathcal{M}}\right)\left(O_{\mathcal{N}} - \omega_m\right) \\ \text{with } h\left(x_1, x_2\right) = \exp\left(-\|x_1 - x_2\|^2 / 2{\sigma_h}^2\right) \end{array} \tag{4}$$

Figure 1 illustrates the use of two maps, $\mathcal{S}$ and $\mathcal{D}$ for sequence processing, as suggested by [5]. $\mathcal{S}$ is a *state map* encoding the state of the sequence. The input of $\mathcal{S}$ is a scalar value $I_{\mathcal{S}} = o \in [0,1]$. At each time step t, $o\left(t\right)$ is taken from a repeated sequence of observations, denoted by letters for the sake of clarity ($A = 0, B = 0.2, C = 0.4, D = 0.6, E = 0.8, F = 1$). For example, using the sequence ABF means that $o\left(t\right)$ takes the values $0, 0.2, 1, 0, 0.2, 1, 0, 0.2, \cdots$ at successive time steps. The context input $C_{\mathcal{S}}$ of $\mathcal{S}$ is the output position $O_{\mathcal{D}}$ of the $\mathcal{D}$ map. This latter map is a *delay map*, that handles the sequential nature of the input by the use of a one time step delay. Indeed the input of $\mathcal{D}$ is $I_{\mathcal{D}}\left(t\right) = O_{\mathcal{S}}\left(t-1\right)$. As for map $\mathcal{S}$, the context input of $\mathcal{D}$ is $C_{\mathcal{D}} = O_{\mathcal{S}}$. In other words, $\mathcal{D}$ represents the one step past output of $\mathcal{S}$, contextualized by its current output, providing the whole architecture with the ability to encode sequences, as shown in further experiments.

Figure 1 shows the three distributions of the matchings for each map. The dark one is the input matching, the one just beside is the context matching, and the middle one the global matching. At time t, the inputs of the two maps are fixed, since $I_{\mathcal{S}}$ is constrained by the current observation $o\left(t\right)$ and $I_{\mathcal{D}}$ by the last $O_{\mathcal{S}}$. The resulting matching distributions $\{\mu_s\}_{s\in\mathcal{S}}$ and $\{\mu_d\}_{d\in\mathcal{D}}$ are thus

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

determined too (they are darkened on Fig. 1). The remaining degrees of freedom of the architecture are the determination of the map outputs, that will condition the remaining matching distributions $\{\nu_s\}_{s\in\mathcal{S}}$, $\{\xi_s\}_{s\in\mathcal{S}}$, $\{\nu_d\}_{d\in\mathcal{D}}$ and $\{\xi_d\}_{d\in\mathcal{D}}$.

The setting of these four distributions is the resonant pathway (see bold arrows in Fig. 1) along which the outputs of the two maps, whose computation has not been presented so far, is determined. This is actually done by iterating the following algorithm until convergence of $(O_\mathcal{S},\, O_\mathcal{D})$

$$\begin{array}{l}\text{compute } \{\nu_s\}_{s\in\mathcal{S}}, \{\xi_s\}_{s\in\mathcal{S}}, \{\nu_d\}_{d\in\mathcal{D}}, \{\xi_d\}_{d\in\mathcal{D}} \text{ from } (O_\mathcal{S},\, O_\mathcal{D}) \\ s^\star \leftarrow \text{argmax}_{s\in\mathcal{S}}\, \xi_s,\ d^\star \leftarrow \text{argmax}_{d\in\mathcal{D}}\, \xi_d \\ O_\mathcal{S} \leftarrow O_\mathcal{S} + \lambda\,(x_{s^\star} - O_\mathcal{S})\,,\ O_\mathcal{D} \leftarrow O_\mathcal{D} + \lambda\,(x_{d^\star} - O_\mathcal{D})\end{array}$$

This fixed point provides the final $(O_\mathcal{S}(t)\,,\, O_\mathcal{D}(t))$ for the current time step. Then, learning can be applied to all the prototypes of the architecture, according to equation (4), and the next time step can be processed from the next observation in the sequence.

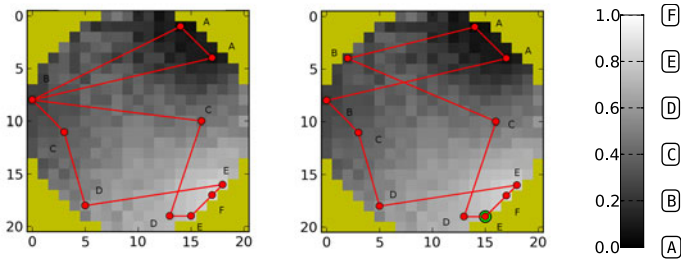
In our experiments, the fixed point is attained in few relaxation steps (about 50), which is much faster than the complex dynamic neural field coupling used in [4, 5]. This is why joint organization can be addressed in the context of efficient machine learning thanks to this shortcut. This will be shown in the following experiments.

3 Results

We run the same experiments as in [5]. In all the experiments, we set $\sigma_\mu = \sigma_\nu = 0.2$, $\lambda = 0.1$, $\alpha = 0.15$, $\sigma_h = 0.06$, $\beta = 0.5$. As in [5], we consider the two sequences $S1 = ABCDEFEDCBA$ and $S2 = ABCBAFEDEF$. These sequences are clearly ambiguous since some observations appear more than once in each of them but in a different context. Only the context in which they appear makes possible the distinction between these inputs. For example in the sequence S1, 'E' is repeated two times but once after a 'D' and once after a 'F'. The input prototypes $\{w_s\}_{s\in\mathcal{S}}$, after 500 time steps, are shown in gray level in Fig. 2(a). As in classical SOM, we observe a continuous mapping of the one-dimensional observations onto the two dimensional map. We also display the locations of the 50 last positions $O_\mathcal{S}(t)$ as a trajectory within the map, as well as the observation presented when a particular unit position was equal to $O_\mathcal{S}(t)$. For sequence S1, each observation appears two times in a different context. After 500 timesteps (figure 2(a)) the sequence is almost learnt. Ten out of eleven elements of the sequence have indeed elicited different $O_\mathcal{S}(t)$ and the ambiguous observations such as 'A' are well-differentiated. The element 'B' of the sequence is however still ambiguous since $O_\mathcal{S}$ is the same the two times it appears in the sequence. It means that the input prototypes are already learnt but the recurrent pathway is not correctly built yet; the observations are differentiated but the temporal sequence is ignored. As the learning process continues, the recurrent pathway organizes itself and the unique $O_\mathcal{S}$ for ambiguous observations split into one unit for each occurrence of this observation in the sequence (cf. Fig 2(b)). This split

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

is typical of what we observe experimentally. The input map usually starts like a classical SOM with the selection of the units depending only on the current observation. As learning goes on, the recurrent map is able to provide a sufficiently informative context to enforce a split within regions of the input map selective for the same observation. Also, we observe (not shown here) that the sequence of $O_{\mathcal{S}}$ within each map is stable.



(a) (b) (c)

Fig. 2: The distribution $\{w_s\}_{s\in\mathcal{S}}$ are shown in gray level after (a) 500 time-steps and (b) 550 time-steps while presenting the sequence $S1 = ABCDEFEDCBA$. The 50 last $O_{\mathcal{S}}(t)$ before these time-steps, as well as the associated observation, are shown along the red trajectory. (c) For clarity, a letter A, B, C, D, E or F is associated with each scalar observations $o = [0, 1]$.

Interestingly, if we now present the sequence $S2 = ABCBAFEDEF$ to the network, the prototypes re-organize. In this experiment, we first present sequence S1 for 700 timesteps. The input prototypes $\{w_s\}_{s\in\mathcal{S}}$ as well as the positions $O_{\mathcal{S}}(t)$ are shown in Fig. 3(a). We then present sequence $S2$. The input prototypes and the positions $O_{\mathcal{S}}(t)$ after 700 additional time-steps are shown in Fig. 3(b). We observe that even if the input prototypes change only slightly, the selected units are completely different. In particular, a striking example is the unit selected when presenting the observation $o = 1.0$ ('F') that became ambiguous when presenting the second sequence. We clearly see that two units are now selected after 1400 time-steps when presenting this observation, depending on the context. Overall, these two experiments illustrate the ability of the architecture to learn a stable representation of a non-markovian sequence and to adapt when a new sequence is presented.

4 Discussion

We have presented an algorithm that allows to implement the resonant competitive processes involved in joint self-organization. This implementation is kept suitable for machine learning applications, avoiding to cope with the complexity

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

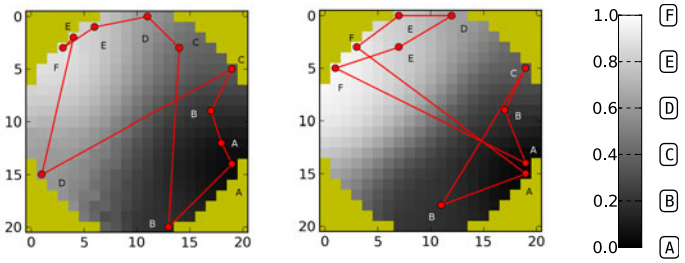


Fig. 3: The distribution $\{w_s\}_{s\in\mathcal{S}}$ are shown in gray level after (a) 700 time-steps while presenting sequence $S1$ (b) 700 additional time-steps while presenting sequence $S2$. As in Fig 2, the 50 last positions $O_{\mathcal{S}}(t)$ are shown.

of coupled neural fields. Our algorithm is stationary, since no decaying learning rates and no shrinking WTA radius is required. Using constant parameters is crucial for preserving the adaptability to react to changes in the input distribution. Another advantage of this approach, compared to the more biologically plausible ones, from which it is derived [4, 5], is the use of the winner positions over the map surface as the only inputs provided by one map to the other. This makes our architecture close to SOM-SD [6, 7] where such reduced information is transmitted in temporal recurrent pathways. Moreover, using winner positions allows to visualize the map-to-map weights organization, and thus better understand the complex inter-map self-organizing processes emerging as a response to the input observation stream. This will help for further studies of the self-organizing dynamics of such architectures.

References

[1] Teuvo Kohonen. *Self Organizing Maps*. Springer, 1997. Second Edition.

[2] Shun-Ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27(2):77–87, 1977.

[3] Lucian Alecu, Hervé Frezza-Buet, and Frédéric Alexandre. Can self-organization emerge through dynamic neural fields computation? . *Connection Science*, 23(1):1–31, 2011.

[4] Olivier Ménard and Hervé Frezza-Buet. Model of multi-modal cortical processing: Coherent learning in self-organizing modules. *Neural Networks*, 18(5-6):646–655, 2005.

[5] Bassem Khouzam and Hervé Frezza-Buet. Distributed Recurrent Self-Organization for Tracking the State of Non-Stationary Partially Observable Dynamical Systems. *Biologically Inspired Cognitive Architectures*, 3:87–104, 2013.

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

[7] Alessandro Sperduti. Neural networks for adaptive processing of structured data. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *ICANN*, volume 2130 of *Lecture Notes in Computer Science*, pages 5–12. Springer, 2001.