Consensus Driven Self-Organization: Towards Non Hierarchical Multi-Map Architectures

Noémie Gonnier, Yann Boniface, Hervé Frezza-Buet

Consensus Driven Self-Organization: Towards Non Hierarchical Multi-Map Architectures

Noémie Gonnier[1] ✉, Yann Boniface[2], and Hervé Frezza-Buet[1]

[1] Université de Lorraine, CentraleSupélec, CNRS, LORIA, F-57000 Metz, France
noemie.gonnier@loria.fr,herve.frezza-buet@centralesupelec.fr
[2] Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France
yann.boniface@loria.fr

Abstract. This paper introduces CxSOM, a model to build modular architectures based on self-organizing maps (SOM). An original consensus driven approach enables to adress non-hierarchical architectures where SOMs get organized jointly. The paper aims at showing how the modules are able to store the association between data, and evaluating, by a mutual information criterion, the resulting organization. These results stand as preliminary work to study bigger architectures.

Keywords: Neural Network Models · Self-Organizing Maps · Multi-Modal Architecture

1 Introduction

Artificial neural networks are an illustration of how computer science can benefit from biologically-inspired paradigms. Deep learning [5] is an example of advances in computer science that has a biological flavor, since it is based on the formal neuron, which is an abstraction of actual nervous cells. Nevertheless, bringing a biological concept up to operative computational techniques usually overcomes the biological side. In deep learning, sophisticated gradient-based techniques and many other refinements do not really have identified counterparts in biology. The present paper follows the same kind of path, from biology to computer algorithms, but rather in the field of self-organizing substrates observed in the cerebral cortex of mammals. The cortex is described as a tiling of similar [14] elementary circuits over a neural surface that get topologically organized by adaptive processes. The cerebral cortex as a whole is understood as an architecture involving numerous interconnected self-organizing modules, handling different modalities and their associations [2, 15]. Self-organizing feature maps (SOMs) introduced by Kohonen [9] model one of such modules. It is nowadays a vector quantization machine learning technique thanks to tricks like argmax computation, decreasing winner-take-all kernel width, decreasing learning rates, that needed to be introduced.

Although the topographical aspect of self-organisation observed on the cortex surface has successfully been transposed to machine learning context, the

modular aspect of the cortical organisation has lesser been explored in this perspective. This motivates the work presented in this paper, since materials and methods are proposed in order to build up modular architectures from self organizing computational modules. On a rather biological side, some models of hierarchical cortically-inspired modular architectures have been proposed [10, 16]. Robotics or computer vision are generally addressed as proofs of concept in such approaches, since the focus is rather on biological plausibility than on the SOM algorithm itself: in [10] the SOM is quite small, in [13] it is used beforehand for initialization and in [16] it is replaced by another vector quantization paradigm close to growing neural gas.

On the machine learning side, some approaches address modular self-organization. A variety of them, reviewed in [17], are based on the ART paradigm, initially inspired from biology. Self-organization addressed in these works is not topographic and are therefore out of the scope of this paper. Approaches involving several self-organizing maps are not necessary a modular architecture either. For example, in [18], the input space is split into several subset so that several SOMs handle a specific subset. Hierarchical SOMs may also be confusing since hierarchy is involved in the computation of a *single* SOM, as clearly explained in [3]. In the end, quite a few works address modular and topographical self-organization. Let us mention [7] which is an algorithmical approach oriented toward letter-phoneme integration as well as A-SOM [8] which introduces associative SOMs. The present work is in line with these two, with a higher stress on computational homogeneity between the modules and scalability for architectures with many modules. The architecture proposed in this paper relies on a dynamical process leading to a global *consensus* among all the modules. This consensus drives the self-organization in each module so that they perform a joint self-organization. This is the extension of previous work involving a cellular paradigm closer to the biological side [12, 11] and leading to an abstraction of the neural field toward SOM-like structures, as initiated in [1]. This paper introduces self-organizing modules as well as a methodology for grouping them into an architecture. Moreover, an original method based on mutual information is introduced to mesure how self-organization arises. The paper is organized as follows. First, a reformulation of the SOM algorithm is introduced, enabling the connection of modules as non-hierarchical architectures and allowing the definition of an original measure of organization. Then experiments on 2D data are presented.

2 Model: CxSOM

The main goal of the study is to set-up rules to build up a SOM architecture which achieves vector quantization tasks, as conventional SOM does, while learning relationship between associated data. The model has to be applicable to any structure and particularly non-hierarchical ones, and unlike A-SOM where some maps are specifically designed to connect other standard SOMs, all the modules are designed to be of the same type, eventually differing by some parameters.

2.1 SOMs as Blocks of an Architecture

Each module i is a slightly modified version of Kohonen's self organizing map [9]. First, let us introduce the standard model: a map is a graph with a fixed size and fixed topology. The input space is noted $\mathcal{D}$, the map being fed by inputs $\xi \in \mathcal{D}$. In this study, for plotting considerations, the graph is a one-dimensional line of N units, where each one is indexed by a position $p = \frac{i}{N-1}, 0 \leq i \leq N$. To each of those units is associated a weight vector $\omega_e(p) \in \mathcal{D}$, also referred to as *prototype*, randomly initialized. The algorithm performs vector quantization by creating a mapping of the input space over the N prototypes, with the specificity that the protoypes of two close units in the map are also close in $\mathcal{D}$, creating a continuity in the mapping. At each learning iteration, a new input $\xi \in \mathcal{D}$ is presented to the map. The BMU is found as the position having the maximal *activity* where

$$a(\xi, p) = \exp[\frac{\|\xi - \omega_e(p)\|^2}{2\sigma^2}] \tag{1}$$

Learning is realized by moving each unit towards ξ relatively to how close it is from Π :

$$\forall p, \omega_e(p, t+1) = \omega_e(p, t) + \alpha \times H_e(\Pi, p) \times (\xi - \omega_e(p, t)) \tag{2}$$

$H(\Pi, p)$ is a linearly decreasing function around Π in map positions space, reaching 0 at a distance h_e. In our approach, the learning rate α and the neighborhood radius h_e are constant, as opposed to most SOM-based works.

2.2 CxSOM Architecture Model

Let us now consider the architecture of n modules connected as a directed graph G. A module (i.e a map) i takes an external input from $\mathcal{D}^i$, and a set of BMU from all the other maps connected to this one. SOM model is then extended to handle these multiple inputs. As G may present cycles, modifiying Π in a map then modifies the activity and thus the BMU of already computed maps; the search for BMU must be a *consensus* between maps.

The notations of the model are summarized in Figure 1 for an example of two maps connected one to another. Let us note ξ the external input of a map, and $\gamma_1, ..., \gamma_K$ its set of contextual inputs which are the BMU Π of all the other maps connected to this one in G. Taking the BMU as the only information transmitted between maps, encoding a state, brings homogeneity in the model: regardless of the dimension of the prototypes of a map, the transmitted information is a position. This encoding has already been used successfully as the information being transmitted between computational steps within a map, in models like SOMSD [6]. It is used here as well as a compact representation of remote map's state. Each unit is associated to $K + 1$ weight vectors, each vector being associated to an input: ω_e is relative to the external input and $\omega_{c1}, \omega_{c2}, ..., \omega_{cK}$ are relative to the contextual inputs. Hence, the contextual weights are mappings over remote map position space. Learning is performed online by presenting a

set of inputs to the architecture. A learning iteration is described in Algorithm 1: inputs are presented to each map in the architecture, and each set of weights ω_{ck} computes its activity distribution a_{ck} from a gaussian matching, relatively to its input. These activities are merge into a global matching a_g.

$$a_g(p) = \sqrt{a_e(\xi, p) \times (\beta a_e(\xi, p) + (1-\beta) a_c(p))}, \quad \text{with} \quad a_c(p) = \frac{1}{K} \sum_{k=1}^{K} a_{ck}(\gamma_k, p) \tag{3}$$

β is a merge factor, set to 0.5 in all the experiments. Notice that a map may have only contextual inputs; its global matching is then a_c, and map activity is driven by the behavior of other maps in the architecture. A spatial gaussian convolution is then applied on a_g, for stability issues not detailed here. Then, a *relaxation* is realized to find a ensemble $(\Pi_0, \cdots, \Pi_N)$ so that each BMU is situated at the maximum of its map's activity . This search is performed by small displacements of Π in each map until a stable state is reached.Once the BMU is found, each set of weights ω_{ck} is separately updated according to Eq.2, relatively to the corresponding input. External neighborhood radius h_e is taken superior to the contextual ones h_c, see figure 1. Equation 3 ensures the global matching is mainly driven by the external inputs, and the difference between neighborhood radius keeps learning process on a smaller and more local scale for contexts compared to external weights. Both properties contribute to learning stability and convergence.

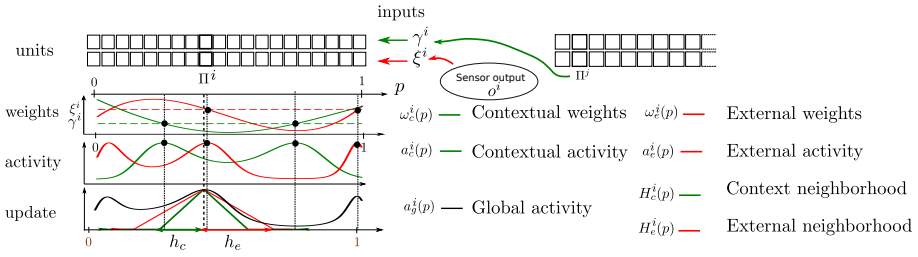

Fig. 1. Notations for a map in a CxSOM architecture, taking one contextual input.

3 Experiments and Results

3.1 Experiments Formalization

In order to exhibit what is learnt during self-organization and to quantify the progression of that learning, the formalization of experiments rely on random variables. Let us consider that the data given to the learning process are sampled independantly and identically (i.i.d.) from a random variable. Here, that

Input: $\xi^1, ..., \xi^K \leftarrow (o^1, \cdots, o^K) \in \mathcal{D}^1 \times \cdot \times \mathcal{D}^K$
$t \leftarrow 0$
$\forall i, \Pi^i \leftarrow \arg\max_p a_e(\xi^i, p)$
while $\mathbf{\Pi}(t) \neq \mathbf{\Pi}(t-1)$ **do**
  forall *Map* i **do**
    $\gamma_1^i, ...\gamma_k^i \leftarrow$ BMUs from connected maps
    Computation of a_g^i (equation 3)
    $p\star^i \leftarrow \arg\max_p a_g^i(p)$
    $\Pi^i \leftarrow \Pi^i + min(\Delta, |p\star^i - \Pi^i|) \times \text{sgn}(p\star^i - \Pi^i)$
  end
  $t \leftarrow t + 1$
end
$\forall i, \omega_e^i(p) \leftarrow \omega_e^i(p) + H_e(\Pi^i, p)(\omega_e^i(p) - \xi^i)$
$\forall i, \forall k, \omega_{ck}^i(p) \leftarrow \omega_{ck}^i(p) + H_c(\Pi^i, p)(\omega_{ck}^i(p) - \gamma^i)$

Algorithm 1: Learning iteration with relaxation process

random variable, denoted by U, does not directly feed the learning process. U is rather "seen" through a set of observation functions $s^1, \cdots, s^n$ so that the input samples are realizations of the joint random variable $(O^1, \cdots, O^n) = (s^1(U), \cdots, s^n(U)) \in \mathcal{D}^1 \times \cdots \mathcal{D}^n$. In our architectural approach, each observation feeds the external input of a dedicated map. As SOMs work online, $(O^1, \cdots, O^n)$ samples are provided one by one, learning being performed at each time step t from the realization $(o_t^1, \cdots, o_t^n)$ of the variable $(O^1, \cdots, O^n)$. Moreover, at time t, each map i computes a BMU Π^i, so that $(\Pi_t^1, \cdots, \Pi_t^n)$ is the state of the whole architecture at t. The tuples $\{(u_t, o_t^1, \cdots, o_t^K, \Pi_t^1, \cdots, \Pi_t^n)\}_t$ can then be viewed as samples of a global joint random variable. Let us complement such tuples with the values $\omega_e^1(\Pi_t^1) \cdots, \omega_e^n(\Pi_t^n)$, that are the input prototypes selected by each map at t. The analysis of the dynamic is performed after last learning step or periodically during learning as follows: learning is freezed temporarily, in order to get a collection of N tuples, from N samples $\{u_n\}_{1 \leq n \leq N}$ of U, subsequent observations and subsequent BMU position within the maps. These samples enable numerical statistical analyses of the random variable of which they are realizations.

Another use of the collected tuples is to evaluate how well a map i has "discovered" the existence of U. If the map has captured the existence of U, its BMU values should be informative about the corresponding U values. The pairs $\{(u_n, \Pi_n^i)\}_{1 \leq n \leq N}$ should draw an actual *function* $u = f(\Pi)$. This is what further plots in figure 4 actually show. In statistical terms, the samples should show that U depends on the random variable Π^i. This dependency can be evaluated from samples by computing *mutual information* between Π^i and U: $I(\Pi^i, U)$, representing in information theory how much information a realization of Π^i carries about U probability. It's defined from entropy H as:

$$I(\Pi^i, U) = H(U) - H(U|\Pi^i) \tag{4}$$

In our results, I is then normalized by $H(U)$, which is the maximum value $I(\Pi^i, U)$ can reaches when u is an actual function of Π. Although this property can be visually evaluated on one dimensional maps and one dimensional data, it gets harder in larger dimensions, in which cases such an indicator is useful. Estimation is realized by discretizing the variables through binning and probabilities estimated by counting the samples. As the datasets are artificial and as big as needed, the estimation is not a problem. In the following, this is performed periodically, while learning takes place, in order to exhibit an increase of the mutual information between Π^i and U as the map gets jointly self-organized. This is illustrated further in figure 5. To sum up, modelling the input as well as the status of the architecture as joint random variables facilitates an illustration and evaluation of what is actually learnt within the maps. This is used further to analyse the experimental setup presented in this paper.

3.2 Evaluation of the Model on Two Maps

This experiment aims at understanding which mecanisms are involved in the organisation by analysing 2 maps connected one to another and therefore developing a methodology to that end. The external inputs (O_1, O_2) are the (X, Y) coordinates of a point on a circle centered in $(0.5, 0.5)$, radius 0.5, both depending on the hidden variable U as $(X, Y) = (\frac{1+cos(2\pi U)}{2}, \frac{1+sin(2\pi U)}{2})$. The architecture is trained on 2000 iterations, U being uniformly drawn in $[0, 1[$ on each one. α is set to 0.1, $h_e = 0.2$, $h_c = 0.07$, $\Delta = 0.01$. After training, learning is freezed and U, thus (X, Y), and (Π^X, Π^Y) after relaxation, are sampled 1000 times in order to build plots in Figs 2, 3, 4. Two samples S_1 and S_2, whose elements are referred to as $(x_1, y_1, u_1, \Pi_1, ...)$ and $(x_2, y_2, u_2, \Pi_2, ...)$ in the following, are highlighted in the plots. They correspond to $x_1 = x_2 = 0.6$, $y_1 = 0$ and $y_2 = 1$, $u_1 = 0.8$ and $u_2 = 0.2$.

The relation ship between $\omega_e^X(\Pi^X)$ and ξ is plotted in figure 3 to evaluate if vector quantization is achieved: it should be close to identity. Even if some accuracy is lost compared to a standard map, vector quantization is correctly achieved within each map. Figure 2 presents the weights repartition in each map after training. It can be seen that the weights are globally disposed according to the external inputs: if two nodes have close external prototypes, then they are close in the map. Within this overall repartition, weights are disposed according to U in a reduced number of "zones", in gray in the figure. For example, S_1 and S_2 share their X value, but have different U. BMUs are thus in two consecutive "zones" on the map according on figure 2, making the distinction between the U value, but keeping the BMU prototype close to the input value. These zones also correspond to the steps in figure 3: as ω_e is smooth, two areas winning for a same interval of X have slightly different ω_e, both still close to X, but corresponding to different U. The right graphic in figure 2 shows on a same plot the center region of map X and the pairs (Π^X, X). It is noticeable that entire zones of map units never win: BMUs are actually located at the extrema of contextual weights, two consecutive zones corresponding to a same range of external inputs, but different U. The context is thus introducing a discontinuity in the map, which

is unexpected in topology preserving self-organization. In figure 4, U value is plotted according to the corresponding BMU position. It shows that U can be deduced from Π in each map.

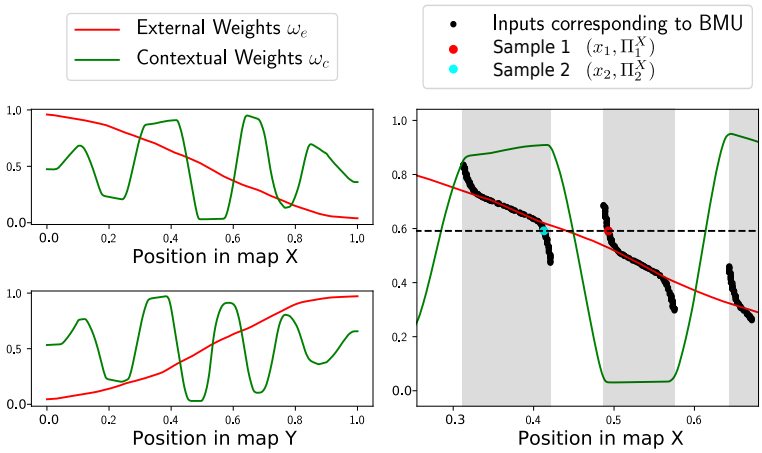

Fig. 2. Weights disposition of a 2-map structure after learning. Inputs corresponding to winning positions on a segment of map X are plotted in the right graphic. The disposition of contextual weights resulting of joint self-organization allows the map to find the BMU for a sample according to U and not only the external input value.

The ability to make a distinction in each map's state depending on U is measured by a mutual information criterion. For sake of comparison between maps which are learning independantly on X and Y and joint maps, figure 5 shows normalized mutual information between U and Π in both maps during learning, as described in section 3.1. Every 20 learning iterations, 5000 samples are computed and mutual information evalutated on them. The curves are a mean of 30 runs of the experiment. Estimation is realized though binning. U bin size is set to 0.02, whereas Π are already discrete variables taking 500 values. To allow comparison, the bin size is the same for the estimation of normalized mutual information on both independant and joint maps. First, the figure shows that mutual information increases from initial random weights to final state; more interestingly, it shows a significant enhancement between independant maps and joint maps. So, while oberving Π in independant maps gives a good precision on X value but lets U uncertain, observing Π in joint maps may lose a bit of accuracy on X but ensures a BMU also carries information about the whole state of observations, and has "discovered" U.

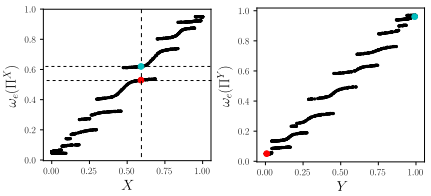

Fig. 3. BMU weight distribution according to inputs. It should be close to identity.

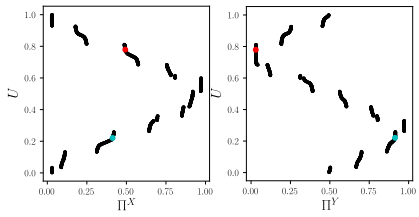

Fig. 4. U distribution according to BMU position. U can be deduced from Π in each map.

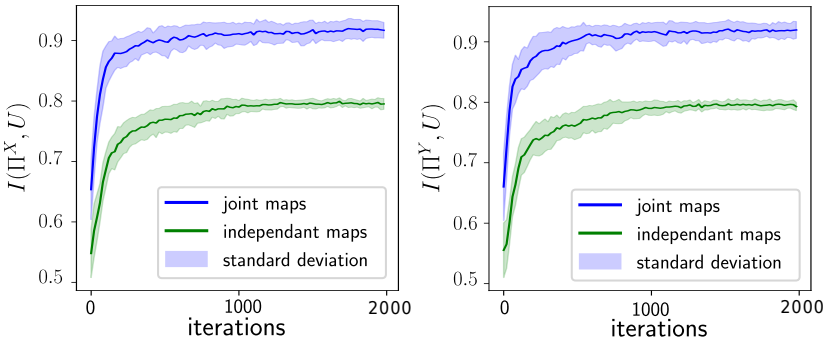

Fig. 5. Mutual information evolution between BMU positions and U.

4 Conclusion

This paper introduces an extension of self-organizing maps usable as module in non-hierarchical architectures. Each of those modules performs vector quantization, as standard SOM does, but the learning process is also driven by a consensus between modules, allowing them to share and store information about remote map's state. This information sharing has been measured by a mutual information based evalutation. As BMUs, i.e positions, are transmitted between maps as representation of a map's state, the model is scalable to any architecture, unrelated to each module's input space dimension. The experiments on two maps depict the organization created by this information transmission: each module organizes itself in regard of its external inputs, but also differentiates the BMUs depending on the global state of observations. This behavior is particularly interesting considering the simplicity of the information shared between maps. This work can be related to the one carried in [4] on recurrent SOM for temporal sequence processing. The basic architecture behavior brought to light in this paper enables forthcoming work on larger architectures, two-dimensional SOMs and integration of recurrent processes.

References

1. Baheux, D., Fix, J., Frezza-Buet, H.: Towards an effective multi-map self organizing recurrent neural network. In: Proc. ESANN'14. pp. 201–206 (2014)
2. Ballard, D.H.: Cortical connections and parallel processing: Structure and function. Behavioral Brain Science **9**, 67–129 (1986)
3. Dittenbach, M., Rauber, A., Merkl, D.: Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. Neurocomputing **48**(1), 199–216 (2002)
4. Fix, J., Frezza-Buet, H.: Look and Feel What and How Recurrent Self-Organizing Maps Learn. In: Proc. WSOM'19, vol. 976, pp. 3–12 (2020)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
6. Hagenbuchner, M., Sperduti, A., Ah Chung Tsoi: A self-organizing map for adaptive processing of structured data. IEEE Trans. Neur. Netw. **14**(3), 491–505 (2003)
7. Jantvik, T., Gustafsson, L., Papliński, A.P.: A self-organized artificial neural network architecture for sensory integration with applications to letter-phoneme integration. Neural Computation **23**(8), 2101–2139 (2011)
8. Johnsson, M., Balkenius, C., Hesslow, G.: Associative self-organizing map. In: Proc. IJCCI'09. pp. 363–370 (2009)
9. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics **43**(1), 59–69 (1982)
10. Lallee, S., Dominey, P.: Multi-modal convergence maps: From body schema and self-representation to mental imagery. Adaptive Behavior **21**(4), 274–285 (2013)
11. Lefort, M., Boniface, Y., Girau, B.: SOMMA: Cortically Inspired Paradigms for Multimodal Processing. pp. 1–8 (2013)
12. Ménard, O., Frezza-Buet, H.: Model of multi-modal cortical processing: Coherent learning in self-organizing modules. Neural Networks **18**(5-6), 646–655 (2005)
13. Miikkulainen, R., Bednar, J.A., Choe, Y., Sirosh, J.: Computational Maps in the Visual Cortex. Springer (2005)
14. Miller, K.D., Simons, D.J., Pinto, D.J.: Processing in layer 4 of the neocortical circuit: New insights from visual and somatosensory cortex. Current Opinion in Neurobiology **11**, 488–497 (2001)
15. Mountcastle, V.B.: The columnar organization of the neocortex. Brain **120**, 701–722 (1997)
16. Parisi, G.I., Tani, J., Weber, C., Wermter, S.: Emergence of multimodal action representations from neural network self-organization. Cognitive Systems Research **43**, 208–221 (2017)
17. Tan, A.H., Subagdja, B., Wang, D., Meng, L.: Self-organizing neural networks for universal learning and multimodal memory encoding. Neural Networks **120**, 58–73 (2019)
18. Wan, W., Fraser, D.: Multisource data fusion with multiple self-organizing maps. IEEE Trans. on Geoscience and Remote Sensing **37**(3), 1344–1349 (1999)