computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

LEARNING TO TEACH TO NEURAL NETWORKS HOW TO LEARN WELL WITH SOH, A SELF-OBSERVING HEURISTIC

Jean-Jacques Mariage

CSAR research group, AI Laboratory, Paris 8 University, 2, rue de la Liberté, St Denis, France, Cdx 93526
jam@ai.univ-paris8.fr

Abstract: *In this ongoing research, we present a Self-Observing Heuristic (SOH). SOH is a hybrid computing method. It roots in natural selection and optimization techniques to provide an environmentally driven evolutionary computation scheme, capable of autonomic cumulative learning. Our aim is to realize an adaptive learning system based on neo-Darwinian evolution of neural units. We proceed in two complementary directions. On one hand, we try to automatically compute the costly tuning phase of the configuration and learning parameters of neural networks (NNs). On the other hand, we use meiosis cellular growth as a natural computation technique to bypass palimpsest effects observed when adding new knowledge to previous one. The main idea is to build an event guided growing competitive NN that develops while it learns to tune other NNs' parameters. Other NNs can be models more or less similar — or even identical — to it. The system adapts itself, learning to teach other models how to learn well.*

Keywords*: – adaptive structure, cumulative learning, emergence, evolutionary architecture, holism, neural networks growth, self-observation.*

1. INTRODUCTION

Most of the current learning algorithms suffer from limitations about evolving capabilities. Self-organizing structures are nevertheless well known processes that, among a large variety of domains, appear in physics and neurobiology [11, 12, 19, 20]. One of the most famous examples is the Belousov-Zhabotinskii reaction. Those processes show that, from particles to galaxies, spontaneous emergence of natural self-organizing structures is quasi universal. Those structures are autocatalytic. More over, they are persistent, resilient and self-propagating for a while, after which they vanish. They rely on a three state cyclic evolution (Active, Inactive, Quiescent (A, I, Q)) and they generate the emergence of complex forms, the characteristics of which are not encoded at the elementary constituent level.

Considering evolutionary learning in terms of self-organization heuristics leads to postulate that connectionist representations are obtained in an ascending constructivist way. Semantic progressively emerges from relations between elementary constituents inside the model on one hand and between the model and the input data manifold on the other hand. A learning system thus appears as a recursive structure relying on two kinds of feedback loops. One kind is internal to the system. The second is external, between the environment and the system.

Learning procedures are usually related to supervised or unsupervised learning. We here focus on the constraints imposed by real-time continuous data in possibly infinite changing and noisy environments (what we call on-line learning) versus delayed data carefully prepared for laboratory experimentation (i.e. off-line learning). Real-time conditions forbid strict supervised learning. On the other hand, techniques like reactive training or reinforcement learning are situated in between supervised and unsupervised learning. They allow a system to pursue a trajectory indicated in the data space by changing the input-target pair. Dual reinforcement learning extends the procedure to two nets, one learning to evaluate the quality of the other's learning.

Data driven programming and error measures are the only way to free the system from the human in the loop. The system can then go its own sweet way, self-supervising by regularly testing its efficiency and reverting to learning mode when necessary.

NNs are very complex processes, which develop their structure in time. From real-time process control and analysis, we can set out that, when faced with too complex processes, usual learning algo-

rithms reach their limit. It thus becomes necessary to resort to more powerful methods. We make the hypothesis that, among the existing NN algorithms, there is a sufficient set of primitives from which a holistic constructive programming scheme [23, 25] can emerge, by means of evolutionary techniques.

We have chosen the Self-Organizing Map (SOM) model [6] because, as attested by a large amount of research publications, the model itself is currently evolving, towards adaptive variants. Moreover, it offers a visual interface of remarkable quality with the data space in an either familiar and accessible bi-dimensional topologically ordered representation. And last but not least, its number of configuration and learning parameters to tune is rather important.

Adaptive topologies are of crucial interest to automatically determine the size of NNs. Oversized nets lead to prohibitive training time, while undersized nets can't find the data space structure. Around back-propagation architectures, efficient growing and pruning methods allow to find a near-optimal number of hidden units. A review has been made in [1]. For SOM, things become more difficult because of topological ordering constraints in the mapping it performs.

The following of this paper is organized in three parts. We first investigate the main properties of the SOM algorithm and its evolutionary growing variants. From that frame, in a second part, we draw out a self-observing heuristic as a minimal system capable of adaptive learning. In the last part, we extract a minimal set of properties, necessary to obtain the emergence of Darwinian evolution among elementary constituents, only constrained locally by a few deterministic rules.

2. SOM AND RELATED EVOLUTIONARY VARIANTS

SOM is a winner take all (WTA) unsupervised learning algorithm. The convergence procedure is iterative. Units are arranged on a regular, usually bi-dimensional, grid. A neighborhood function $N(.)$, guided by the lattice, acts as a dissipative structure. It spreads neural activation from the best match unit to a surrounding active area. This mechanism is responsible for a local ordering process, which is the key to a global convergence of the algorithm to an ordered solution [5, 8]. Various shapes exist for $N(.)$. N_{width} (the neighborhood size) regularly decreases as a function of time. The intensity of spreading activation can be a function of inter-units distance [21] or uniform [6].

As regularly reminded by Kohonen [5, 8], SOM learning is divided into two training stages. In a first (crude) self-organization phase, clusters of units find their place and density (reflecting their intra and inter relations). In a second (fine tuning) phase, the algorithm converges on the basis of the previously extracted structure of the data classes. To obtain an ordered mapping of good quality, a few conditions are necessary. The initial N_{width} must be very large in the early training phase[1], and stop with one ring of neighboring units at the end of the process. Kohonen [7, p. 4913] states that "*The clustering effect is more complex in the case where the width of the kernel is much smaller than the width of the network*". In the convergence phase, N_{width} can be significantly reduced or eventually kept constant. Parameters need to be manually tuned for the two training phases. The initial size of the neighborhood considerably increases the cost of the learning procedure. It thus becomes of prime importance to precisely evaluate the number of units.

To try to overcome these problems, growing variants of SOM have been designed, either constrained on a geometrical structure, or free on a graph. The construction of the map space then becomes an evolutionary process, only driven by the data distribution. We won't describe those architectures in details here, for they are now well known and frequently used (for a survey, see [13 or 14]). They bring the considerable advantage of automatically specifying a very delicate interlinked set of parameters[2].

Despite this, some of them present major drawbacks. Many parameters remain to be manually specified. Some variants even introduce additional ones, such as the maximum number of units allowed [2, 3, 4, 24], connection and disconnection thresholds [2], a fixed number of adaptation steps, a required precision [4] or a growing speed [22], that seem at least as tricky to operate as the standard model's parameters. Except for [22], the growth process needs to be controlled by means of the usual trial and error intuitive procedure.

SOM performs a dimensionality reduction from an n-dimensional input data space into an internal bi-dimensional memory space. Reduction provides efficient abstraction for the resulting mapping. The counterpart is that neighboring clusters can be grouped together as a consequence of an inappropriate dimension selection for the map instead of reflecting an implicit relation detected in the structure of the data manifold. For high dimensional data spaces with unknown distribution, such misclassification problems are not detectable.

Unlike SOM, a part of its evolutionary growing variants provide a mapping that is isomorphic to the

1 *i.e.* covering at least 30% to 50% of the total number of units.
2 That is the shape and the dimension of the map, its number of units, and sometimes even the type of lattice, which determines the neighborhood density.

data space. They are exact Topology Representing Nets (TRN) [17], and thus, the mapping obtained suffers from a lack of abstraction and topological ordering loss. A few variants [2, 6] keep the grid lattice to drive the growth process in a bi-dimensional space. They maintain the convenient visualization interface offered by SOM and improve it by visualizing the frontiers of the data classes. Those, the structure of which is free on a graph loose either the topological ordering and the visualization.

However, experimentation with growing variants of SOM have led to a gradual suppression of the global parameters in evolutionary topological map algorithms and to localize the whole set of parameters to the unit.

3. SETTING THE FRAMEWORK

SOH relies on a dual learning procedure between NNs. It is similar to the Dual Heuristic Programming (DHP) [10], but with three main differences. First, the only assumption we make about the actor model is that we need a standard unsupervised primitive extractor model (SOM, Adaptive Resonance Theory (ART), Neo-Cognitron, etc.) Second, the critic net is an SOM-based evolutionary (incremental and adaptive) algorithm. Third, we try to use the actor-critic-actor loop to automatically extract, encode, and make use of learning primitives detected in the actor model.

The system evaluates the quality of the learning process other NN models are performing. To let it look at the way NNs learn, we give it the time course of their parameters as context vectors like in the temporal SOM or in RNNs. The associated data are organization measures of the observed models. While SOH evolves, its structure develops to reflect the universe it is exposed to. Its world being learning algorithms, defined by the relation between their parameters and organization measures, it thereby learns to identify good learning processes. In turn, since it learns to appreciate the quality of learning in other models by considering the time course of their configuration parameters, it learns by the same way to tune the learning parameters of basic models. Furthermore, as SOH learns to teach to classifiers that work as extractors of primitives too, it learns to extract the learning invariants it finds in other models. Once trained, SOH is able to recover the context from the data and vice versa.

4. BASIC CELL PROPERTIES

Constraints on intercellular activity are limited in number. They settle through afferent and efferent connection beams. A weight vector figures the incoming synapses of the cell, or its receptive field. Outgoing synapses are a dissipative structure that spreads activation in the neighborhood. In order to capture possible hierarchical relations between parameters, SOH is tree structured. We explicitly endow the cell with a quiescent state. Its activation function has three possible states: Active, Inactive, Quiescent (A, I, Q), necessary to obtain the emergence of self-replicating and resilient forms. Cellular death prevents from overcrowding. The selection of the active element results from competition. All the units, provided they are not in quiescent state, have equal chances to be selected. At every moment, there exist a finite number of active cells. Internal and external retroaction loops make the structure recursive and store a trace of the past states in memory. The whole configuration parameter set is localized at the unit level.

Insertion and suppression heuristics control the system growth and equilibrium. The process is very brittle [2, 3, 4]. It must find a compromise between stability and plasticity. The various ways mentioned in the literature to deal with this problem rely on the activation frequency of the units and on a distance measure between their weight vectors.

Duplication emanates from a representation conflict. A cell duplicates when an activity bud develops, indicating a local perturbation, where relations with the environment exert a pressure. The bud shows that too much activity accumulates over the same cell and claims for its repartition (the resource redistribution according to B. Fritzke). Three kinds of perturbation are mainly used in the SOM-based evolutionary architectures. Growing Cell Structure (GCS) considers the triggering frequency, Bungy-SOM the sum of the triggering errors, while Incremental Grid Growing (IGG) and Dynamic Cell Structure (DCS) use the sum of the distances to the entries that trigger the cell. Depending on the case, a new cell connects to all the neighbors of the cell it splits from, to the single error node (IGG), or between the most frequently activated unit, and the neighboring cell with the most distant memory (GCS). The new cell inherits from the common sensibility to the activity that induces this outgrowth. Its memory is initialized as if it had been part of that cell cluster since the beginning of learning, with the mean of the memory vectors of all its neighbors. What we call meiosis duplication. The new cell thus grabs a trace of the sensitivity to the whole features gathered in the neighborhood tree. The splitting cell (the old one) becomes inactive for a while before turning to the quiescent state, from which it gradually recovers sensibility. This transitory state creates an idle time, which drives the activity transfer onto the new cell. The cell specializes by cooperative and competitive interaction with the related elements. It connects to, or reinforces its connections with, any cell sensible to the same activity. It separates from,

or decreases the strength of its connection with, any element sensible to a different activity.

In the literature two main ways of implementing the intermediate refractory state are adopted. Either there exist an explicit disposition in the activation rule, or the balance between excitation and inhibition processes is assumed to implicitly induce the same effect. In the former, a quiescent cell becomes active again after several selection attempts during its resting time, as in the GCS. In the latter, inhibition is considered to be equivalent to a kind of non-local refractory state. Transition between the states A, I, Q is continuous. A brief absolutely quiescent period immediately follows the active state. The next state is a relatively refractory phase where cell's excitability is reduced. That is the quiescent state succeeds those two ones. The competitive version of the Hebb rule, due to Martinetz [15], to compute the lateral connections strength, builds up such a mechanism.

$$C_{ij}^{(t+1)} = \begin{cases} y_i.y_j & \text{if} \quad y_i.y_j \geq y_k.y_l, \quad 1 \leq k,l \leq N, \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where $y_i = R\left(\| x - w_i \|\right)$ is the response of the i^{th} unit for an input vector x. $R(.)$, the mapping function, $R \rightarrow [0, 1]$, is continuous, and monotonically decreasing.

In case of non-stationery data distributions, the rule is modified to let the connection strength decrease until it vanishes. A decay constant α, and a suppression threshold θ, such as $0 < \alpha$, $\theta < 1$ are introduced.

$$C_{ij}^{(t+1)} = \begin{cases} 1 & \text{if} \quad y_i.y_j \geq y_k.y_l, \quad 1 \leq k,l \leq N, \\ 0 & \text{if} \quad C_{ij}^{(t)} < \theta, \\ \alpha\, C_{ij}^{(t)} & \text{otherwise} \end{cases} \qquad (2)$$

For off line learning procedures, with a learning set T of size $|T|$, α takes the value

$$\alpha = \sqrt[|T|]{\theta} \qquad (3)$$

The cell is endowed with a lifetime. It is the frequency of its exchanges that keeps it alive. It can be the node activation frequency as well as triggering conflicts or neighborhood commitments. Units that have not been selected for a (parametrical) while degenerate and disappear. Every cell manages its own neighborhood, which evolves by its own. It has its own learning rate. One cell can learn faster than another or it can not learn at all. Most of the architectures examined in [13 and 14] depend on a regular lattice of units and fix those two parameters by keeping them constant. Fritzke's GCS, [4] maintains one ring of neighbors and two different learning rates, one for the best match unit, and another for the neighbors.

The unit has a tree-structured neighborhood, allowing it to capture multiple influences with complex structure (lists, trees, graphs). Neighborhood role is crucial. It must establish kernels of connections to serve as suitable bases for unit clusters to develop and specialize. To be efficient, the chosen function must allow clusters to develop dense connectivity. The global growth of the system requires a neighborhood either of variable size and reduced in the early learning. On one hand, clusters encode categories or functional abilities. They must structure according to the density of the data forms they represent and to their affinities with the other groups. On the other hand, the system gradually builds itself from one — or a few — unit(s) that split when needed. The evolution of the neighborhood cannot follow the particularly efficient regular decreasing scheme T. Kohonen adopted for the SOM, which remains the prime example as far as neighborhood is concerned.

We thus propose a connective neighborhood, local to the unit, based on an algorithmic dynamical dissipative structure relying on a free graph. To allow a saw tooth evolution of the neighborhood, trees of sub-trees (algorithmically Steiner partial minimal spanning trees, hereafter denoted STs) progressively grow or shrink and connect or disconnect on the fly, during learning. An ST is the optimal diffusion tree. The algorithm adds new branching nodes to the neighborhood tree — not to the underlying graph — and deletes costly edges to optimally cover relational sub-manifolds on graph structures. The resulting tree is built in such a way that it been an MST of parametric depth and cardinality.

We denote a graph G as a set of vertices V and edges E.

$$G = (V, E) \text{ tq. } \forall v_i \in V \; \exists \, v_j \in V, (v_i, v_j) \in E \qquad (4)$$

The vertices are points in a vector space, defined on $\mathbb{R}^n$. Learning can be considered as a graph matching process between different spaces. The learning system (the network) is a graph in the weight space, where n-dimensional memory vectors represent the coordinates of the nodes. This graph, internal to the net, must develop its structure in order to reflect the graph of the relations that exist in the data space.

The distance between two points a and b is expressed by an Lp metric denoted $\|a-b\|_p$ where p denotes the chosen distance metric.

In the rectilinear ($L1$) metric,

$$\| a - b \|_1 = \| x_a - x_b \| + \| y_a - y_b \|.$$

In the Euclidean ($L2$) metric,

$$\| a - b \|_2 = \sqrt[2]{(x_a - x_b)_2 + (y_a - y_b)_2} .$$

The length of an edge $e = (a - b)$ is $\|e\| = \| a - b \|$.

The length of a tree, Γ is $\|\Gamma\| = \sum_{e \in \Gamma} \|e\|$.

We consider a non oriented graph $G = (V, E)$, and $d_{weights}(.)$, a non negative cost function, expressed in an arbitrarily chosen metric on the edges of G. For every subset S of vertices of G, a Steiner tree T for S is a sub-tree of G connecting all the vertices in S. The cost of the ST is the sum of the individual costs of its edges. An ST is minimal if its cost is minimal among all the STs.

An ST presents very interesting properties. First, it provides automatic creation of neighboring nodes, to supply the system needs, even trees to connect forests. Second, in case of unlimited number and complexity of the connection structure, computation of the exact partial ST is NP-complete. We must thus resort to approximate cost reducing methods, thereby introducing fuzziness in the learning process and possibly compensate the drawback of exact topology representing nets using the competitive Hebb rule. Third, the tree being parametric, it allows a saw-toothed evolution of the neighborhood size, which appears to be a fundamental condition for topology ordering preservation on growing nets. Finally, when topological ordering is under way, the process is not very brittle. It is not necessary to recompute the neighborhood every time step. The periodicity can be drawn by the system itself, considering an ordering measure of the topology. A comparison made by D. Polani [18] shows that the μQH measure provides acceptable results.

5. ALGORITHM DESCRIPTION

The literature mentions two ways to implement the internal loop, which memorizes anterior states. Passed activity is incorporated to the actual activity by mean of either a context input vector or dual basic cells.

The external loop is an adaptive process inspired by Darwinian natural selection. Units creation and suppression mechanism drives the net towards adaptation to the data evolution. The cooperation and competition schema completes it. From the unit to the whole structure, as well as for clusters and substructures that make it up, the constituents permanently struggle for survival. A higher-level loop is related to error measure. By that way, the system is exclusively events driven programmed.

To prevent from the intrusion of any omniscient entity, learning is un- or self-supervised. The structure grows freely, driven by the data distribution. Events that generate change in the error evolution regulate alternation between learning and tests.

Error allows to verify the quality of the convergence, automate the tuning of some parameters and to find a stopping criterion. With certain algorithms, a global error is enough (statistical methods, MLP). Others (SOM) require a more sophisticated error measure because of the neighborhood links.

Local error points out an environmental pressure, which generates an accumulation of activity. It indicates that some data forms compete around conflicting representations. Either the same unit attracts too many forms, or several units are sensitive to the same form. Data vectors compete because, on one side, they share enough common features, but on the other side, they remain sufficiently different for being attracted by various prototypes.

Global error evaluates the convergence quality. This estimation can be used to determine a stopping test or to automatically tune configuration parameters as the learning rate or the neighborhood size.

A sudden increase of the quantification error indicates a rarely met state. The observing system can then automatically re-enter in learning mode.

Learning ends when the criteria chosen as stopping test no more significantly evolves. In realistic learning simulation conditions, the system must alternate learning trials over finite clean data sets — in order to create basic receptive structures — as well as over potentially infinite noisy data streams, applied in real time.

With finite learning bases, the simplest test compares the average error E, over the input data vectors set at the current iteration, to a threshold. The error is computed between the i^{th} input and the j^{th} triggering unit memories.

$$E^{(t)} = \frac{1}{X} \sum_{i=1}^{X} \underset{j}{\operatorname{argmin}}\, d\left(\boldsymbol{x}_i, \boldsymbol{w}_j^{(t)}\right), \quad j = 1, \cdots, N \tag{5}$$

where X is the learning vectors number, N the units number, and $d(.)$ a distance measure, expressed in a chosen metric.

The stopping criterion S is a threshold, chosen arbitrarily small. It is compared to the ratio between the error measures, at the current time iteration (t), and at the previous one (t-1).

$$\frac{E^{(t-1)} - E^{(t)}}{E^{(t)}} < S \tag{6}$$

For a more accurate observation, the average error decline is considered since the last node creation. New nodes are added if the error does not decrease quickly enough during a sliding time window [1].

$$\frac{E^{(t)}-E^{(t-tw)}}{E^{(t^*)}} < \delta_T \tag{7}$$

where *tw* is the time window width, *t** the iteration following the last node creation, and δ_T is a user defined trigger slope parameter. The process can be automatically settled by setting δ_T to half the smallest inter-errors difference.

$$0 < \delta = min\left\{\left|s_i - s_j\right|\right\}, \tag{8}$$

with $\left|s_i - s_j\right| > 0$ and $1 \le i, j \le N$. The error being considered as insignificant when

$$\sum_{i=1}^{n} \left\| x_i^{(t-1)} x_i^{(t)} \right\| < \frac{\delta}{2} \tag{9}$$

x^i being the activation state of the ith unit.

In SOM, the learning rate, $\alpha^{(t)}$, $0 < \alpha^{(t)} < 1$ is a global parameter. It decreases monotonically as a function of time. A linear evolution in 1 - (t / T_{total}) is generally chosen, t being the current iteration index, and T_{total} the number of time steps. Kohonen et al. [9] suggest, as a second choice, an inverse function of time in C / (C + t). C being a constant with a value of $C = T_{total} / 100$. We adopted the same scheme, with T_{total} being the width of the current sliding time window.

The neighborhood radius self-tunes. Any new cell j_{new} sets the initial size of its neighborhood. Among all the other cells, units the activation state of which is lower than a threshold are part of its neighborhood. Possible thresholds are the mean of the activation values or the median value.

$$N(j_{new}) = \forall\, j \in G \;\middle|\; Act(j) \le \frac{1}{N}\sum_{j=1}^{N} Act(j),\; j \ne j_{new} \tag{10}$$

The amount of time elapsed since the last neighborhood update defines a confidence parameter. It linearly decreases in $1/N_{local\_it}$, where N_{local_it} is the number of time steps. Old neighborhoods must be recomputed. In [5], tests carried out with a minimal spanning tree neighborhood, show that a recomputing frequency every 10 to 100 iterations is enough. A cell creation thereby initializes a neighborhood that is large with regard to the number of units, and sets its confidence to 1.

The probability density of the regions in the data space is generally unknown. It corresponds to the receptive field of the cell — its triggering stimuli — *i.e.* to the Voronoi region in the data space the center of which is the current triggered cell. It can be estimated by the cell's triggering frequency (Fritzke [4], p. 1446 & foll.). Beyond bi-dimensional universes, computing the Voronoi tessellation becomes very complex. It is however possible to "*estimate the volume of the Voronoi regions by the mean length of the edges* $\bar{l}_c$ *emanating from the cell c in an n-dimensional hypercube*".

As stated in [4], the neighborhood is the estimate volume of the Voronoi cell, *i.e.* the mean length of the edges.

$$\bar{l}_c = 1 / \, card(N_c) \sum_{i \in N_c} \left\| \boldsymbol{w}_c - \boldsymbol{w}_i \right\| \tag{11}$$

Remotest nodes from the best match unit are deleted from the active neighborhood, not from the map. We simply compare their activation values as:

$$\frac{Act(j) - Act(bmu)}{Act(bmu)} \tag{12}$$

The number of vertices in the neighborhood, *card*(*Nc*(.)), thereby automatically increases or decreases. One way to heighten the saw tooth evolution of the neighborhood is to affect the slope of the function by mean of a cooling schedule ranging between the mean and half the mean.

When a cell becomes the best match unit its activation frequency $\tau_{bmu}^{(t)}$ is incremented. For all the other units, it is decremented by an amount of δ, $0 < \delta < 1$.

$$\tau_{bmu}^{(t+1)} = \tau_{bmu}^{(t)} + 1 \tag{13}$$

$$\tau_i^{(t+1)} = \tau_i^{(t)} - \delta \,.\, \tau_i^{(t)}, \quad \forall i \ne bmu \tag{14}$$

New units' weight vectors are initialized according to :

$$w_{new} = \frac{1}{card(N_c+1)} \left(\boldsymbol{x} + \sum_{i \in N_c(new)} \boldsymbol{w}_i \right) \tag{15}$$

where $card(N_c(new))$ is the neighborhood of the cell from which the new cell splits from.

Following [22], the learning procedure is adapted to on-line conditions. The learning step $\alpha^{(t)}$ and the neighborhood radius $h_{bmu,i}^{(t)}$ are no longer defined, as in SOM, as a function of the learning time (which can be unknown). They are related to the cells' activation frequency τ_i . Those parameters, which are global in SOM become local to the cell.

The learning rule thus becomes:

$$\boldsymbol{w}_i^{(t+1)} = \boldsymbol{w}_i^{(t)} + \alpha_{bmu,i}^{(t)}\, h_{bmu,i}^{(t)} \left(\boldsymbol{x} - \boldsymbol{w}_i^{(t)} \right) \tag{16}$$

6. ALGORITHM SCHEMA

Initialize the first cell with: its activation state $s_0 = 1$, memory = the first data vector, its activation frequency $\tau_{new} = 1$, and learning rate.

Do

- Apply an input vector $\boldsymbol{x}$ on the captors.
- *Competition*: select the best match unit *bmu*
- Adapt activation frequencies with (13) and (14).
- If there exist too *old neighborhoods*:
  - list cells that re-compute their neighborhoods.
- If *conflict*,
  - *Insertion* of a new cell:
  - set its activation frequency $\tau_{new} = 1$,
  - set its weights as in (15)
  - Append it to re-computed neighborhood list.
- Else
  - apply learning rule (16) to *bmu* and its neighborhood.
- If there are *obsolete cells*,
  - delete them and their links with direct neighbors.
- Re-compute *neighborhood*s for listed cells.

While the *stopping criterion* is not satisfied

Competition: Only active nodes are allowed to compete. We locate *bmu* among them. If there exist several *bmu* units, we randomly select one among them. The *bmu* unit is the root of a neighborhood tree. It stores the mean distance[3] between the active units and the data vector as the automatic neighborhood membership threshold θ. Units with an activation that falls under θ's value are possible neighbors.

Conflict: a conflict is detected when more than one cell triggers for the same input.

Insertion: no learning takes place when an insertion occurs. We only compute the new cell's neighborhood.

Neighborhood: gather cells according to (10).

Stopping criterion: learning ends when the stopping criterion as defined in (6) and (7) is reached.

Obsolete cells: cells are deleted if their activation frequency τ_i is less than a given Threshold.

7. CONCLUDING REMARKS

We presented a simple heuristic as a general theoretical framework to apply evolutionary NNs to the design of automatic parameterization. This can settle a basis from which SOH can further be extended to automatic experiments in order to integrate a more complete set of learning primitives. NNs performance on specific tasks strongly depends on node type(s) and learning rule(s) choice. Human design of the learning process notoriously relies on past experience with similar models and contexts. Once the power of evolution is set in motion, it becomes very interesting to study how SOH classifies, recombines, and so on… the learning features it extracts. Such a system would allow to automatically merge together various algorithms or selected parts of them in modular systems and to train them.

8. SELECTED REFERENCES

[1] T. Ash, and G. Cottrell (1995). "Topology-modifying neural network algorithms". In Michael A. Arbib, ed., *Handbook of Brain Theory and Neural Networks*, MIT Press, 990-993.

[2] J. Blackmore, R. Miikkulainen (1993). "Incremental Grid Growing: Encoding High-Dimensional Structure into a Two-Dimensional Feature Map". *Procs. of the IEEE ICNN*, San-Francisco, CA.

[3] J. Bruske and G. Sommer (1995). "Dynamic cell structure learns perfectly topology preserving map". *Neural Computation*, 7, 845-865.

[4] B. Fritzke (1994). "Growing Cell structures – a self-organizing network for unsupervised and supervised learning". *Neural Networks* 7, (9), 1441-1460.

[5] J. Kangas, T. Kohonen, J. Laaksonen, O. Simula, and O. Ventä (1989). "Variants of self organizing maps". *Procs. of the IJCNN'89*, II, 517-522.

[6] T. Kohonen (1982). "Self-Organized formation of topologically correct feature maps". *Biological Cybernetics*, 43, 59-69.

[7] T. Kohonen (1989). *Self-Organization and Associative Memory*. Springer Series in Information Sciences, Third. Edition, Springer-Verlag, Berlin.

[8] T. Kohonen (1993). "Things you haven't heard about the Self-Organizing Map". *Procs. of the IJCNN'93*, 1147-1156.

[9] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, (1995). "SOM_PAK: The Self Organizing Map program package". Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science.

3 or the value of the median distance.

[10] G. Lendaris, and C. Paintz (1997) "Training Strategies for Critic and Action Neural Networks in Dual Heuristic Programming Method". *Procs. of the IEEE ICNN'97,* 712-717.
[11] B. F. Madore, and W. L. Freedman (1983). "Computer simulations of the Belousov-Zhabotinsky reaction". *Science*, 222, 437-438.
[12] B. F. Madore, and W. L. Freedman (1987). "Self-organazing structures". *American Scientist*, vol. 75, N° 3, 252-259.
[13] J-J. Mariage (2000). Architectures neuronales évolutives, un état de l'art. RR CSAR 00-12-01-17, Laboratoire d'IA, université Paris 8.
[14] J-J. Mariage (2001). De l'Auto-Organization à l'Auto-Observation. Ph.D. Dissertation, Department of Computing Science, AI Laboratory, Paris 8 University.
[15] T. Martinetz (1993). "Competitive hebbian learning rule forms perfectly topology preserving maps". In Stan Gielen and Bert Kappen, Eds., *Procs. of the ICANN'93*, 427-434.
[16] T. Martinetz, and K. Schulten (1991). "A neural gaz network learns topologies". In T. Kohonen et al. (Eds.), *IEEE ICNN'91,* 1, 397-407.
[17] T. Martinetz, and K. Schulten (1994). "Topology representing networks". In *Neural Networks*, 7(3), 505-522.
[18] D. Polani, (1997). "Organization mesures for Self-Organazing maps". *Procs. of WSOM'97*, 280-285.
[19] I. Prigogine (1980). *From being to becoming: time and complexity in the physical sciences*. Freeman.
[20] I. Prigogine, I. Stengers (1984). *Order out of chaos: Man's new dialogue with nature*. Bantam.
[21] H. Ritter, and K. Schulten (1988). "Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements". In *Neural Computers*, R. Eckmiller and C. von der Malsburg Eds., Springer-verlag, 393-406.
[22] T. Trautmann, T. Deneux (1995). "Comparison of dynamic feature map models for environmental monitoring". *Procs. of the ICNN'95,* I, 73-78.
[23] F. Varela, E. Thompson, and E. Rosch (1993). *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press, Cambridge, MA.
[24] TH. Villmann, H.-U. Bauer (1997). "The GSOM-algorithm for growing hypercubical output spaces in self-organizing maps". *Procs. of the WSOM'97,* 286-291.
[25] T. Ziemke (1999). *Rethinking Grounding*. In Riegler, Peschl, von Stein (Eds.), Understanding Representation in the Cognitive Sciences, New York: Plenum Press.

Jean-Jacques Mariage *was born in Saisseval, France, in 1953. While working from 1973 to 1999 in postal and telecommunication civil service, he began studying computer science at the University of Paris 8 in 1990 where he obtained his PhD in 2001. He joined the artificial Intelligence Lab. inside the CSAR research group in 1994 where he now pursues his research as a post PhD. His work on automatic parameterization tuning of unsupervised NN models led him towards his today's main interest which is holistic programming of biologically inspired adaptive systems. His current interests include NN algorithms, evolutionary programming, artificial life, learning, memory, evolution, and biological aspects of encoding structures development.*