INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

A survey of fuzzy clustering algorithms for pattern recognition

A. Baraldi*, P. Blonda†

TR-98-038

October 1998

Abstract

Clustering algorithms aim at modelling fuzzy (i.e., ambiguous) unlabeled patterns efficiently. Our goal is to propose a theoretical framework where clustering systems can be compared on the basis of their learning strategies. In the first part of this work, the following issues are reviewed: relative (probabilistic) and absolute (possibilistic) fuzzy membership functions and their relationships to the Bayes rule, batch and on-line learning, growing and pruning networks, modular network architectures, topologically perfect mapping, ecological nets and neuro-fuzziness. From this discussion an equivalence between the concepts of fuzzy clustering and soft competitive learning in clustering algorithms is proposed as a unifying framework in the comparison of clustering systems. Moreover, a set of functional attributes is selected for use as dictionary entries in our comparison. In the second part of this paper, five clustering algorithms taken from the literature are reviewed and compared on the basis of the selected properties of interest. These networks clustering models are: i) Self-Organizing Map (SOM); ii) Fuzzy Learning Vector Quantization (FLVQ); iii) Fuzzy Adaptive Resonance Theory (Fuzzy ART); iv) Growing Neural Gas (GNG); and v) Fully self-Organizing Simplified Adaptive Resonance Theory (FOSART). Although our theoretical comparison is fairly simple, it yields observations that may appear

*International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94704-1198, Ph.: +1+510+643-9153, Fx.: +1+510+643-7684, baraldi@icsi.berkeley.edu

†IESI-CNR, Via Amendola 166/5, Bari 70126, Italy, Ph.: +39+80+5481612, Fx: +39+80+5484311, blonda@iesi.ba.cnr.it

parodoxical. Firstly, only FLVQ, Fuzzy ART and FOSART exploit concepts derived from fuzzy set theory (e.g., relative and/or absolute fuzzy membership functions). Secondly, only SOM, FLVQ, GNG and FOSART employ soft competitive learning mechanisms, which are affected by asymptotic misbehaviors in the case of FLVQ, i.e., only SOM, GNG and FOSART are considered effective fuzzy clustering algorithms.

Key words: Probalistic and possibilistic fuzzy membership, on-line and batch learning, modular architectures, topologically correct mapping, ecological nets, fuzzy clustering.

1 Introduction

In recent years the synthesis between clustering algorithms and fuzzy set theory has led to the development of several so-called fuzzy clustering algorithms whose aim is to model fuzzy (i.e., ambiguous) unsupervised (unlabeled) patterns efficiently. The goal of this paper is to review and compare self-organization strategies of clustering algorithms that have been called fuzzy, or can be considered fuzzy, according to some definitions found in the existing literature. The best of our knowledge, few such comparative studies have been attempted (e.g., [1]). This may be due to the objective difficulty of comparing the great variety of clustering approaches based on a meaningful set of common functional features. To select a set of interesting functional features, in the first part of this paper the concepts of relative and absolute fuzzy membership functions, batch and on-line learning, growing and pruning networks, modular network architectures, topologically correct mapping, ecological net and neuro-fuzziness are reviewed. Selected functional attributes are employed as dictionary entries in the comparison of clustering algorithms. We derive our own interpretation of fuzzy clustering inductive learning, intended as a synonym of soft competitive parameter adaptation in clustering systems, from the existing literature. This conceptual equivalence is employed as a unifying framework in the comparison of clustering algorithms.

This approach has been considered "interesting" and "quite reasonable" by some researchers [2]. However, other authors believe that since "fuzziness can be incorporated at various levels to generate a fuzzy neural network, i.e., it can be at the input, output, learning or neural levels" (see [3], where each input feature is expressed in terms of fuzzy membership values indicating a degree of belonging to each of the linguistic properties *low*, *medium* and *high*), our "claim of calling certain networks to be fuzzy/nonfuzzy seems improper and not acceptable" [2]. In both cases, comments pertaining to the proposed terminology "fuzzy clustering algorithm" do not affect the core of this work, which is centered on the comparison of learning mechanisms adopted by several clustering algorithms to model fuzzy (ambiguous) patterns.

In the second part of this paper, the basic functional features of five clustering algorithms are investigated to to see if these algorithms: a) employ, to any degree, fuzzy set-theoretic concepts, such as absolute and relative membership functions, fuzzy set operations, granularization, etc.; and/or b) belong to our conceptual framework of fuzzy clustering systems. The clustering models investigated are:
i) on-line learning, static-sizing, static-linking Self-Organizing Map (SOM) [4], [5];
ii) off-line learning, static-sizing, no-linking Fuzzy Learning Vector Quantization, FLVQ [6] (which was first called Fuzzy Kohonen Clustering Network, FKCN [7]);
iii) on-line learning, dynamic-sizing, no-linking Fuzzy Adaptive Resonance Theory (Fuzzy ART) [19], [20];
iv) on-line learning, dynamic-sizing, dynamic-linking Growing Neural Gas (GNG) [26], [27]; and v) on-line learning, dynamic-sizing, dynamic-linking Fully self-Organizing Simplified Adaptive Resonance Theory (FOSART) [28], [29], based on the Fuzzy Simplified Adaptive Resonance Theory (Fuzzy SART) model [24], [25].

2 Fuzzy membership and probability density functions

This section proposes a brief review of probabilistic and possibilistic fuzzy membership concepts, to be compared with Bayes' view of posterior probability and likelihood.

2.1 Absolute and relative fuzzy memberships

Let X_k be instance k of the input manifold X, where $k = 1, ..., n$, such that n is the total number of input instances. Let us assume that X_k may belong to a generic *state* (also termed *category* or *component*) C_i, $i = 1, ..., c$, where c is the total number of possible states. The extent to which X_k is compatible with a vague (fuzzy) concept associated with generic state C_i can be interpreted "more in terms of a *possibility (compatibility) distribution* rather than in terms of a *probability distribution*" [30], p. 58. This legitimizes some possibility distributions, called fuzzy membership functions, that "we believe are *useful*, but might find difficult to justify on the basis of objective probabilities" [30], p. 57. Depending on the conditions required to state that c fuzzy states C_i, $i = 1, ..., c$, are a fuzzy c-partition of the input data set, membership functions can be divided into two categories [61], [62]:

1. *relative or probabilistic or constrained fuzzy membership values* $R_{i,k}$; and

2. *absolute or possibilistic fuzzy membership (typicality) values* $A_{i,k}$,

where index k ranges over patterns and index i over concepts. Absolute and relative membership types are related by the following equation:

$$R_{i,k} = \frac{A_{i,k}}{\sum_{h=1}^{c} A_{h,k}}, \quad i = 1, ..., c, \quad k = 1, ..., n. \tag{1}$$

Relative typicality values, $R_{i,k}$, must satisfy the following three conditions [7], [30]:

(i) $R_{i,k} \in [0,1]$, $i = 1, ..., c$, $k = 1, ..., n$;

(ii) $\sum_{i=1}^{c} R_{i,k} = 1$, $k = 1, ..., n$; and

(iii) $0 < \sum_{k=1}^{n} R_{i,k} < n$, $i = 1, ..., c$.

Constraint (ii) is an inherently probabilistic constraint [61], relating $R_{i,k}$ values to posterior probability estimates in a Bayesian framework. Because of condition (ii), $R_{i,k}$ values are relative numbers dependent on the absolute membership of the pattern in all other classes, thus indirectly on the total number of classes. This also means that Processing Elements (PEs) exploiting a relative membership function as their activation function are context-sensitive, i.e., $R_{i,k}$ provides a tool for modeling network-wide internode communication by assuming that PEs are coupled through feed-sideways (lateral) connections [64].

Although possibilistic membership functions, $A_{i,k}$, $i = 1, ..., c$, $k = 1, ..., n$, may satisfy conditions (i) and (iii) listed above (in this case, the upper bound of the membership function is one and the fuzzy set is termed normal [30]), they always differ from probabilistic memberships in condition (ii), which is relaxed as follows [61]

(iv) $\max_i \{A_{i,k}\} > 0$, $k = 1, ..., n$.

Owing to condition (iv), the sum of absolute memberships of a noise point in all the "good" categories need not be equal to one. Term $A_{i,k}$ is an absolute similarity value depending on fuzzy state C_i exclusively, given input pattern X_k. In other words, $A_{i,k}$ is context-insensitive, since it is not affected by any other state. Thus, PEs exploiting an absolute membership as their activation function are independent, i.e., they feature no lateral connection.

Both probabilistic and possibilistic fuzzy clustering are affected by some well-known drawbacks. On one hand in probabilistic fuzzy clustering, owing to condition (ii), noise points and outliers, featuring low possibilistic typicalities with respect to all templates (codebooks), may have significantly high probabilistic membership values and may severely affect the prototype parameter estimate (e.g., refer to [62]). On the other hand, in possibilistic fuzzy clustering, learning rates computed from absolute typicalities tend to produce coincident clusters [62], [63]. This poor behavior can be explained by the fact that cluster prototypes are uncoupled in possibilistic clustering, i.e., possibilistic clustering algorithms try to minimize an objective function by operating on each cluster independently. This leads to an increase in the number of local minima.

Different $A_{i,k}$ expressions in the existing literature and consistent with the definition provided above were found to be useful. These include the following:

$$A_{i,k} = \begin{cases} \frac{1}{1+d_{i,k}^2/\eta_i} \in (0,1] \text{ [61]}, & (2) \\ Gaussian_{i,k} = e^{-\frac{d_{i,k}^2}{2\sigma_i^2}} \in (0,1] \text{ (Gaussian mixtures; [66], [67])}, & (3) \\ \frac{1}{(d_{i,k}^2)^{p_i}} \in (0,\infty) \text{ [58]}, & (4) \\ \frac{1}{(1-Gaussian_{i,k})^2} \in (1,\infty) \text{ [25]}, & (5) \end{cases}$$

where $d_{i,k} = d(X_k, T_i)$ is assumed to be the Euclidean distance between input pattern X_k and prototype (receptive field center) T_i of the i-th category. Variables σ_i, η_i and p_i are all resolution parameters belonging to range $(0, \infty)$ (see [62]).

2.2 Fuzzy memberships and mixture probabilities

To investigate the relationship between (objective) probability density functions and (useful) fuzzy membership functions, note that absolute membership function (3) relates probabilistic membership (1) to Gaussian mixture models, which are widely employed in the framework of optimization problems featuring a firm statistical foundation [9], [31], [32], [43]. In a mixture probability model consisting of c mixture components C_i, $i = 1, ..., c$, let $p(C_i)$ be the *a priori* probability that a pattern belongs to mixture component C_i, and $p(X_k|C_i)$ the conditional likelihood that the pattern is X_k, given that the pattern's state is C_i. If these statistics are known, *a posteriori* conditional probability $p(C_i|X_k)$ can be estimated using Bayes' rule as

$$p(C_i|X_k) = \frac{p(X_k|C_i) \cdot p(C_i)}{\sum_{h=1}^{c} p(X_k|C_h) \cdot p(C_h)}, \quad k = 1, ..., n, \quad i = 1, ..., c. \tag{6}$$

If $p(C_h) = 1/c, \forall h \in \{1, c\}$, i.e., all states are assumed to be equally likely, then Equation (6) becomes

$$p(C_i|X_k) = \frac{p(X_k|C_i)}{\sum_{h=1}^{c} p(X_k|C_h)}, \quad k = 1, ..., n, \quad i = 1, ..., c. \tag{7}$$

The following relationships hold true:

(i) $p(C_i|X_k)$, $p(X_k|C_i)$ and $p(C_i)$ belong to range $[0,1]$;

(ii) $\sum_{h=1}^{c} p(C_h|X_k) = 1$, $k = 1, ..., n$, i.e., mixture components C_i, $i = 1, ..., c$, provide a complete partition of the input space; and

(iii) $\sum_{h=1}^{c} p(C_h) = 1$.

From the comparison of Equation (1) with Equation (7), and of properties (i)-(iv) in Section 2.1 with properties (i)-(iii) above, we can write that,

when priors are considered the same (i.e., they are ignored), since $\{p(X_k|C_i)\} \subset \{A_{i,k}\}$, thus, $\{p(C_i|X_k)\} \subset \{R_{i,k}\}$; in other words, *when priors are ignored, (objective) likelihood and posterior probabilities are a subset of (useful) absolute and relative fuzzy membership functions respectively.*

To summarize, the combination of Equation (1) with constraints (i)-(iv) in Section 2.1 allows the human designer to choose any absolute membership function that, in addition to satisfying the mild condition (iv) of Section 2.1, is considered useful for the application at hand, although this choice may be difficult to justify on the basis of objective probabilities (see Section 2.1). If the chosen absolute membership function satisfies, not only condition (iv) of Section 2.1, but the more severe constraint (i) above, then absolute membership values are equivalent to likelihood values; as a consequence, relative membership values computed with Equation (1) can be considered posterior probability estimates in the case in which priors are ignored.

3 On-line versus off-line model adaptation

All inductive learning systems, i.e., learning systems progressing from special cases (e.g., training data) to general models, are based on the following concepts: i) information extracted from a finite set of observed examples, termed *training data set*, can be used to answer questions either about unobserved samples belonging to a so-called *test set*, or about unknown properties hidden in the training data; and ii) the goal of the learning process is to minimize a risk functional (theoretically computed over an infinite data set) by adapting system parameters on the basis of the finite training set [35], i.e., the learning problem is turned into an optimization problem [39].

When system parameters are learned from training data, there are two classes of learning situations, depending on how data are presented to the learner: the "batch" setting in which data are available as a block, and the "on-line" setting in which data arrive sequentially [39]. In many practical problems, when a sequential data stream can be stored for analysis as a block, or a block of data can be analyzed sequentially, the user is free to take either the batch or the on-line point of view [39].

The goal of on-line learning methods is to avoid storage of a complete data set by discarding each data point once it has been used [33]. On-line learning methods are required when: a) it is necessary to respond in real time; b) the input data set may be so huge that batch methods become impractical, because of their numerical properties (see below), or computation time, or memory requirement; and c) the input data comes as a continuous stream of unlimited length which makes it totally impossible to apply batch methods [33], [39], [43]. On-line learning typically results in systems that become order-dependent during training (in line with biological complex systems [44]). Moreover, on-line systems, where parameter adaptation is example-driven [42], are more sensitive to the presence of noise as they do not average over the noise on the data, i.e., they tend to provide highly oscillatory (non-smooth) output functions that perform poorly in terms of generalization ability. For example, clustering systems like Fuzzy ART, where a single poorly mapped pattern suffices to initiate the creation of a new unit, may be affected by overfitting, i.e., the system may fit the noise and not just the data.

Batch methods are preferred when our only interest is in a final answer, i.e., the best answer that we can obtain from a finite training data set as the exact closed form solution to a minimization problem [39]. In batch learning problems (e.g., the simple case of linear model regression, see [33], p. 92, and [39]), exact closed form solutions can lead to numerical difficulties for very large data sets. In these cases the only computationally feasible alternative is provided by iterative batch methods, such as the gradient descent of the cost function [33], [42], that sweep repeatedly through the data set. To summarize, batch learning methods are subdivided into: i) exact closed form solutions to the cost function minimization problem; these solutions are numerically and/or computationally inapplicable for very large data sets; and ii) iterative batch learning algorithms, such as the gradient descent of the cost function.

In iterative batch learning algorithms, the learning rate parameter must be small and its choice is fairly critical: if the learning rate is too small the reduction in error will be very small; if it is too large instabilities (divergent oscillations) may result [33]. Analytically, when convergence of iterative batch algorithms to exact closed form solutions is analyzed, then useful hints on constraint of the learning rate value are gathered [39].

Although iterative batch learning algorithms are developed to process very large data sets, they can spend an enormous amount of computation time in processing the entire training set (e.g., in order to compute the gradient), but they may end up by taking a small learning step in the parameter space after each processing epoch. If this is the case (e.g., for gradient descent algorithms), this functional feature is clearly incompatible with the original motivation that justifies the study of such iterative batch learning schemes. In such situations, an alternative solution, called "mini-batch" [39], is to average parameter update values over subsets of the entire training data set. By taking intermediate steps in the parameter space, iterative mini-batch algorithms may converge faster than their iterative full batch counterparts. By stretching the same idea further, another alternative solution to iterative full batch algorithms is to develop their on-line (stochastic)[1] approximations [33].

On-line learning algorithms are simple and intuitive because they are based on the following heuristic ([33], p. 46; [40]): the sum over the training samples, which is found in the iterative and batch solution of the cost function minimization task, is dropped by separating the contribution from the last data point to provide a sequential update formula, i.e., to allow one parameter update for every new data point presentation. Although this heuristic seems reasonable, there should be some analytical proof that the on-line proce-

[1] "Stochastic" refers to the assumption that the single data point being presented to the learning system is chosen on the basis of a stochastic process [39].

dure converges to a solution. It is known that the difference between exact batch-mode and heuristic on-line updating is arbitrarily reduced by the adoption of "small" learning rates [42], [48]. According to the view that on-line procedures are approximations of iterative batch algorithms, learning rate constraints capable of guaranteeing convergence of the iterative batch mode may be applied to the on-line problem under an appropriate definition of convergence (for the linear regression case, see [39]). If these conditions hold, it has been observed that on-line learning systems, by requiring significantly less computation time per parameter update, can be significantly faster in converging than iterative batch algorithms [39].

In general, the learning rate $\alpha(t)$ of the on-line update rule must satisfy the three conditions applied to the coefficients of the Robbins-Monro algorithm for finding the roots of a function iteratively, which are (see [33], pp. 47, 96):
i) $\lim_{t \to \infty} \alpha(t) = 0$;
ii) $\sum_{t=1}^{\infty} \alpha(t) = \infty$;
and iii) $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$.
For example, in an on-line update procedure, if the learning rate remains fixed, then the algorithm converges only in a stochastic sense [39], i.e., model parameters drift from their initial positions to quasi-stationary positions where they start to wander around in dynamic equilibrium [27]. When the learning rate decreases monotonically under Robbins-Monro conditions (e.g., $\alpha(t) = 1/t$, see [33], p. 96, and [27]), on-line learning algorithms can be shown to converge to a point in the parameter space [39]. As a brief review of batch update and on-line update techniques, refer to [27].

4 Beyond error gradient descent: advanced techniques for learning from data

In recent years the neural network community has made a considerable effort in the search for learning techniques that are more effective in dealing with local minima and generalization to new examples than the traditional approaches based on simple gradient descent. These alternative approaches have to deal effectively with the *curse of dimensionality* and with the qualitative principle known as *Occam's razor*. As an example of the curse of dimensionality, consider that any function estimator increases its number of adjustable parameters with the dimensionality of input space. As a consequence, the size of training data required to compute a reliable estimate of adaptive parameters may become huge in practical problems [33], [35].

The complexity of a learning system increases with the number of independent and adjustable parameters, also termed degrees of freedom, to be adapted during the learning process. According to the qualitative principle of Occam's razor, a sound basis for generalizing beyond a given set of examples is to prefer the simplest hypothesis that fits observed data [33], [42]. This principle states that to be effective, the cost function minimized by an inductive learning system should provide a trade-off between how well the model fits the training data and *model complexity*. This also means that model complexity must be controlled by *a priori* (background) knowledge, i.e., subjective knowledge available before any evidence (e.g., empirical risk) provided by the training data is observed. Different inductive principles provide cost functions considered as different quantitative formulations of Occam's qualitative principle [35].

A rough taxonomy of advanced techniques for optimal learning, originally proposed in [40], is presented hereafter.

4.1 Global optimization

Instead of local algorithms like gradient descent, one may explore techniques that guarantee global optimization while effectively facing the curse of dimensionality. Among the most significant developments in this area, Support Vector Machines (SVMs), based on the Vapnik-Chervonenkis (VC) statistical learning theory and capable of detecting the global minimum of a cost function for classification problems, are becoming increasingly popular in finding solutions to both classification and function regression tasks [35], [36], [37].

4.2 Growing networks and pruning

Human designers typically have the opportunity to embed task-specific prior knowledge in an inductive learning algorithm, e.g., by setting the topology and the complexity of a multilayer perceptron when backpropagation weight adaptation is applied. Pruning algorithms begin training a network expected to be large with respect to the problem at hand, and then continue by pruning nodes that do not affect the learning process significantly [40], [42]. Vice versa, growing networks start from small networks that grow gradually until convergence is reached [27], [40]. As a result, the complexity of the network is expected to be tuned to the problem at hand, i.e., generalization capability is expected to increase.

4.3 Modular architectures: prior structures and experience-based fine-tuning

Self-organizing neurological systems consist of highly structured, hierarchical architectures provided with feed-back mechanisms [45]. In these systems, the combination of an initial architecture produced by evolution and experience-based additional fine-tuning prepares the whole system to function in an entire domain by generalizing its learned behaviour to instances not previously encountered [46].

In line with biological learning systems, a classical engineering paradigm consists of partitioning the solution to a problem between several modules specialized in learning a single task, i.e., modular architectures are the natural solution to most significant practical problems [40], [44]. In applied mathematics, the principle of tackling a problem by dividing it into simpler subproblems whose solutions can be combined to yield a solution to the complex problem is termed *divide and conquer* [38].

In supervised learning, an interesting modular proposal that addresses the major problem of providing effective integration of the system modules is presented in [38].

Analytically, the importance of developing modular architectures has been stressed in [40], [41], where sufficient (but not necessary) conditions capable of guaranteeing local minima free cost functions are detected, such that a simple gradient descent algorithm can always reach the absolute minimum of the error surface.

Contiguous to the problem of fine-tuning modular learning systems on the basis of training experiences is the problem of prior structures, i.e., the problem of learning from tabula rasa [40]. Minsky claims that a"significant learning at significant rate presupposes some significant prior structure" [47]. In other words, important properties of the model must be "hard-wired or built-in, perhaps to be tuned later by experience, but not learned in any statistical meaningful way" [71].

Intuitively, starting from some prior learning structure, experience-based (inductive) fine-tuning of network parameters should allow a structured organization of a distributed system to emerge naturally from elementary interactions of PEs. This is tantamount to saying that competitive adaptation of distributed (localized: neuron-based, synapse-based) parameters is expected to enhance the development of structured nets consisting of specialized subsystems (modules), in line with biological neural systems.

With regard to the exploitation of distributed parameters, it can be observed that, on the one hand, most of the on-line clustering algorithms presented in the existing literature, e.g., SOM [4], exploit a *global* (network-based) time counter, i.e., a time variable that is not specialized on a localized basis, rather than *distributed* (localized: neuron-based, synapse-based) time variables. As a consequence, at a given processing time, plasticity (i.e., the potential ability of moving a template vector toward an input pattern) is the same for every PE, in these networks. On the other hand, in recent years, several on-line Kohonen-based network models exploiting distributed time variables have been presented [24], [26], [27], [28]. For example, in GNG and FOSART a connection-based time variable is equal to the number of times one synapse is selected at adaptation steps. In Fuzzy SART and FOSART a neuron-based time variable is equal to the number of epochs that PE has survived. To summarize, on-line clustering algorithms found in the literature depend to different degrees on distributed parameters whose competitive adaptation is consistent with the development of structured systems (i.e., specialized subsystems).

With regard to batch and iterative algorithms, such as Fuzzy $c-$means (FCM) and FLVQ [6], [7], global time rather than distributed time variables are employed, i.e., at a given processing time, plasticity is the same for every PE in the net. This is justified by considering that when fuzzy clustering mechanisms are employed, all PEs acquire the same input pattern simultaneously. However, one may consider that fuzzy membership functions allow a pattern to belong to multiple categories to different degrees. In other words, in batch algorithms the plasticity of every PE is considered as being the same, even though learning "histories" of PEs may differ significantly. This becomes obvious if, for example, for every PE we define a distributed (neuron-based) time counter (equivalent to an *inverse plasticity* or *stability* variable) as the sum of the neuron learning rates at adaptation steps, such that the learning rate of each PE decreases monotonically with its local time. To summarize, in batch clustering algorithms found in the literature, exploitation of distributed rather than global parameters may deserve further investigation in the framework of developing self-organizing networks consisting of specialized subsystems.

5 Topologically correct mapping

The presentation of the Competitive Hebbian Rule (CHR) [49], introducing competition among synaptic links, represented a fundamental breakthrough in the evolution of Fully

Self-Organizing artificial Neural Network models (FSONN). In this paper, a synaptic link is defined as a lateral connection between two PEs belonging to the same neural layer, and FSONN models are defined as those distributed systems capable of: i) dynamically generating and removing PEs; and ii) dynamically generating and removing synaptic links.

CHR generates synaptic links as follows. For a given pattern X_k, $k \in \{1, n\}$, where n is the total number of input patterns, let us consider: a) winner unit PE_{w1} as the one featuring the shortest inter-pattern distance $d(X_k, T_{w1}) \leq d(X_k, T_i)$, $i = 1, ..., c$, where c is the total number of PEs, T_{w1} is the template pattern of PE_{w1}, and $d(X_k, T_i)$ is the Euclidean inter-pattern distance between X_k and T_i; and b) the second best unit, PE_{w2}, as the one featuring activation value $d(X_k, T_{w2}) \leq d(X_k, T_i)$, $i = 1, ..., c$, $i \neq w1$. The exploitation of the Euclidean inter-pattern distance in competitive learning shapes neuron receptive fields as Voronoi polyhedra [49]. According to CHR, if connection between PE_{w1} and PE_{w2} does not exist, it is generated.

It has been proved that CHR forms Topology Preserving Maps (TPMs) [49]. To define a TPM, let us consider an input manifold $X \subseteq R^D$, where D is the dimensionality of input space, and a graph (network) G, consisting of vertices (neural units) PE_i, $i = 1, ..., c$, such that a pointer $T_i \in X$ (also termed category template or prototype), belonging to the pointer set $\{S\} = \{T_1, ..., T_c\}$, is related ("attached") to vertex PE_i. Given S, mapping ϕ_S from input space X onto the vertices of G is defined as

$$\phi_S : X \rightarrow G, \quad X_k \in X \rightarrow w1(X_k) \in G, \tag{8}$$

where X_k is a feature vector (input pattern), $k \in \{1, n\}$, and the vertex $PE_{w1(X_k)}$, also termed winner unit, is determined by

$$d(X_k, T_{w1(X_k)}) \leq d(X_k, T_{i(X_k)}), \quad \forall i \in G, \tag{9}$$

where $d(\cdot)$ is the Euclidean inter-vector distance. According to Equations (8) and (9), a feature vector X_k is mapped to vertex $PE_{w1(X_k)}$, the pointer $T_{w1(X_k)}$ of which is closest to X_k. This is equivalent to stating that X_k is mapped to vertex $PE_{w1(X_k)}$ whose Voronoi polyhedron $V_{w1(X_k)}$ encloses X_k. The Voronoi polyhedron of a neuron PE_i is the receptive field of the PE centered on T_i, and it is identified as $V_i \subseteq R^D$. The masked Voronoi polyhedron of neuron i is defined as $V_i^{(X)} = V_i \cap X \subseteq X$ [49].

Inverse mapping ϕ_S^{-1} from G onto X is defined as

$$\phi_S^{-1} : G \rightarrow X, \quad i \in G \rightarrow T_i \in X. \tag{10}$$

Two pointers T_i, T_j are termed adjacent on the feature manifold X if their masked Voronoi polyhedra are adjacent, i.e., if $V_i^{(X)} \cap V_j^{(X)} \neq 0$. Two vertices PE_i, PE_j in G are termed adjacent if they are connected by a synaptic link. Mapping ϕ_S from X to G is defined as neighborhood (adjacency) preserving if any pair of adjacent pointers T_i, T_j on X are assigned to vertices PE_i, PE_j that are adjacent on G. Mapping ϕ_S^{-1} from G to X is defined as neighborhood preserving if any pair of vertices PE_i, PE_j that are adjacent on G are assigned to adjacent locations T_i, T_j that on X.

Thus, a TPM is defined as a mapping ϕ_S from X to G such that ϕ_S, together with inverse mapping ϕ_S^{-1} from G to X, are neighborhood (adjacency) preserving [49].

Note that exploitation of CHR in FSONNs allows generation of networks consisting of mutually disjointed (specialized and independent) maps [27], [49]. This modular organization enables the learning system to perform, both *cooperative learning* by adapting concertedly those processing units that are connected within graph G (i.e., those units that belong to the same map in the graph), and competitive learning among disjointed maps to enhance specialization of the system's modules.

6 Artificial cognitive systems and ecological nets

To perform cognitive tasks, biological neural systems exploit: i) dishomogeneous nets, where several types of PEs are combined; ii) structured architectures, consisting of hierarchies of subnets; and iii) feed-back mechanisms, where feed-back information is provided by the external environment to the natural system in response to the system's actions [50]. It is *the presence of this feed-back interaction with the environment that characterizes all natural systems featuring cognitive capabilities* [44]. Some artificial neural systems feature none of the biological properties listed above. For example, SOM [4] and the Hopfield network [52] are homogeneous systems; they feature no structured architecture and no *supervision* or *reinforcement* by, or *feedback* from, an external environment (also termed *supervisor*). In *reinforcement learning*, the neural system is allowed to react to each training case. It is then told whether its reaction was effective or not [53]. To increase their biological plausibility, artificial neural models should employ differentiated structures provided with dishomogeneous layers, specialized subnets, hierarchies of maps, etc. In parallel, the study of artificial neural nets as stand-alone systems should evolve to become the science of *ecological nets* (econets), where neural systems as well as their external environments are modeled [50]. For example, unlike Kohonen's networks [5], an ART system employs a structured architecture to self-adjust the network dimension to problem-specific conditions. In particular, the ART *orienting subsystem* models the responses of the external environment to the learning activities of the *attentional subsystem* [19], [20], [21], [22]. Thus, an ART system belongs to the class of ecological nets.

7 On fuzzy clustering algorithms

A clustering algorithm performs unsupervised detection of statistical regularities in a random sequence of input patterns.

Our attention is focused on fuzzification of clustering learning schemes. In the definition presented in [8], it is stated that an artificial Neural Network (NN) model performs *fuzzy clustering* when it allows a pattern to belong to multiple categories to different degrees depending on the neurons' ability to recognize the input pattern. This approach is well known in the traditional field of coding techniques for data compression. Since the traditional k-means clustering algorithms feature a cost function (discretization error) characterized by many local minima, data compression techniques modify c-means algorithms by replacing their *Winner-Takes-All* strategy (WTA), also termed crisp or hard competitive, with a "soft-max" adaptation rule [9], hereafter referred to as soft competitive learning.[2]

[2]Note that terms soft-max and soft competitive learning rule adopted in this paper are not to be confused

A WTA parameter adaptation strategy is purely competitive and allows no cooperative (soft competitive) learning. It is sensitive to initialization of templates, i.e., different initializations may lead to very different minimization results. In fact, WTA adaptations may not be able to get the system out of a poor local minimum when this lies in the proximity of the status where the system was started [27]. Unlike WTA learning, a *soft competitive learning* scheme is defined as a learning strategy that not only adjusts the winning cluster but also affects all cluster centers depending on their proximity to the input pattern [9]. In general, soft competitive learning decreases dependency on initialization and reduces the presence of dead units [27]. We observe that:

1. The fuzzy clustering definition provided in [8] is equivalent to the definition of the soft competitive adaptation rule traditionally employed in the field of data compression [9], i.e., a clustering algorithm is termed *fuzzy clustering algorithm* iff it employs a soft competitive (non-crisp) parameter adaptation strategy.
2. As a corollary of point 1. above, a fuzzy clustering algorithm does not necessarily exploit concepts derived from fuzzy set theory such as fuzzy set membership functions and fuzzy set operations. For example, SOM pursues soft competitive learning by means of biologically plausible update rules that employ no fuzzy set-theoretic concept. Another interesting example is the one provided by the Expectation-Maximization (EM) algorithm applied to optimize parameters of a Gaussian mixture [31], [33]. Although it features a firm statistic foundation and employs no fuzzy set-theoretic concept, EM applied to Gaussian mixtures can be termed fuzzy according to definition 1. presented above.

It is to be noted that relative or probabilistic fuzzy membership functions are traditionally applied to clustering algorithms in the intuitive belief that "vector quantizers based on both winner and non-winner information about the relationship of an input pattern to the prototypes will be better representatives of the overall structure of the input data than those based on local information alone" [6]. Thus, several clustering algorithms, such as the Fuzzy $c-$means (FCM) and the Fuzzy Learning Vector Quantization (FLVQ) algorithms, combine local and global information in the computation of a relative fuzzy membership function [6], [7]. From a functional standpoint, connectionist models where a "useful" relative fuzzy membership function or an objective posterior probability estimate is computed (e.g., FLVQ and EM applied to Gaussian mixtures, respectively), are equivalent to distributed systems where a contextual (competitive and cooperative) effect mechanism employing feed-sideways (intra-layer) connections is employed (see Section 2). Note that only few clustering networks employ intra-layer connections explicitly, i.e., by means of specific data structures and parameter adaptation strategies [26], [27], [28], [29].

with the so-called soft-max function or normalized exponential employed in mixture models [33], and mixture-of-experts [38].

8 Dictionary entries in the comparison of clustering algorithms

From the general discussion developed in Sections 2 to 7 we derive the following set of basis features for use as dictionary entries in a theoretical comparison of data clustering algorithms:

- fuzzy set-theoretical concepts, such as absolute and relative membership functions (see Section 2).
- On-line, batch and mini-batch learning modes, which apply to global (network-based) as well as local (neuron- and connection-based) parameters in distributed learning systems (see Section 3).
- Prior structure, modular architectures, growing and pruning distributed systems (see Section 4).
- Topologically correct mapping, this concept being related to that of modular architectures consisting of specialized and disjointed (mutually independent) maps, which belong to an output lattice of processing units (see Section 5).
- Ecological nets (see Section 6).
- Our own interpretation of a fuzzy clustering algorithm, intended as a clustering system exploiting soft competitive (i.e., cooperative and competitive) versus hard (crisp, purely) competitive adaptation of system parameters (see Section 7).

9 SOM

Starting from Kohonen's on-line Vector Quantization model (VQ, an acronym used herein in line with [4]), which employs a WTA learning strategy, on-line SOM develops a soft competitive learning technique based on two constraints, derived from neurophysiological studies and providing an annealing schedule [9], [64], [65]. They consist of two empirical functions of time, which are user-defined and which obey two heuristic rules [4], [5], the first of which requires learning rates to decrease monotonically with time according to a cooling scheme, i.e., as the number of input pattern presentations increases, all learning rates (winner as well as non-winner) must decrease towards zero (in line with the Robbins-Monro theorem, see Section 3). Important properties of this cooling schedule have been analyzed in [10], [27], [33], [35], [58], [59].

The second heuristic rule applies to the output lattice of processing units and requires the size of the update (resonance) neighborhood centered on the winner node to decrease monotonically with time, such that a soft competitive learning strategy changes into a hard competitive (WTA) one. This model transition is equivalent to stating that the initial overlap (oversampling [11]) between nodes' receptive fields must decrease monotonically with time until it is reduced to zero, as hard competitive learning renders receptive fields equivalent to Voronoi polyhedra [27]. Interpretations of this second heuristic rule, and relationships between SOM and other optimization techniques such as "maximum-entropy" clustering

[12], deterministic annealing [13], and the Expectation-Maximization (EM) optimization algorithm [31], are discussed in [9], [14], [15], [32], [35], [43]. From a general perspective, it is to be remembered that compared to hard competitive learning soft competitive learning not only decreases dependency on initialization, but also reduces the presence of dead units (see Section 7).

Finally, it is important to point out that a batch and iterative SOM version has also been presented [5], [34].

9.1 Input parameters

To simplify the discussion, our approach is to deal with a finite training data set $\{X\}$, consisting of n input patterns X_k, $k = 1, ..., n$, where $X_k \in \mathcal{R}^D$, such that D is the dimensionality of the input space. The input data set is repeatedly presented to the network until a termination criterion is satisfied.

• SOM employs an output array of PEs whose size c is fixed on an *a priori* basis (i.e., the human designer has to make this design choice based on prior knowledge of the particular learning task).
• The dimensionality of the output lattice (1-D to 3-D) must be set by the human designer. This choice implicitly fixes the number of adjacent vertices, which is equivalent to fixing the number of synaptic links emanating from each PE of the output lattice (see Section 5). For example, each PE belonging to a 1-D lattice features two adjacent vertices, PEs belonging to a 2-D lattice feature eight adjacent vertices each, etc. This property of SOM is termed *static-linking*. The peculiar feature of SOM is that it deals with the topological relationship of node adjacency in graph G without dealing with synaptic links explicitly, as shown in Equation (11) (see Section 9.2).
• It assumes that the initial position of clusters is known in advance. These initial positions can be randomly assigned or may be driven by sample vectors extracted from the input data set. The latter solution is likely to reduce the formation of dead units [27].
• It employs two user-defined monotone decreasing functions of time to compute: i) step size $\epsilon(t)$ (*cooling schedule*, see Section 3); and ii) size $\sigma(t)$ of the resonance domain centered on the winner PE in the output lattice. These two decreasing functions are further combined (multiplied) to compute the lateral excitatory signal generated by the winner and featuring the shape of a "bell curve" (see Equation (11)). Time t is the number of input pattern presentations.
• It requires a termination threshold, e.g., the maximum number of epochs e_{max}, which is to say the number of times the finite training data set is repeatedly presented to the network.

9.2 Description of the algorithm

• Geometrical (distance) relationships in the input (measurement) space are employed to enforce competitive learning mechanisms: for an input pattern $X(t) = X_k(t)$, $\forall k \in \{1, n\}$, where n is the total number of patterns and t is the number of input pattern presentations, winner unit $PE_{w1(t)}$ is detected as the processing unit featuring the shortest inter-pattern distance $d(X(t), T_{w1(t)}(t)) \leq d(X(t), T_i(t))$, $i = 1, ..., c$, where $T_{w1(t)}(t)$ is the template pattern of winner unit $PE_{w1(t)}$, and $d(X(t), T_i(t))$ is the Euclidean or Hamming inter-pattern distance. The exploitation of the Euclidean inter-pattern distance in competitive learning shapes neuron receptive fields as Voronoi polyhedra (see Section 5).

• Topological relationships among PEs belonging to an output lattice (graph, see Section 5) are employed to detect neighboring (cooperative, soft competitive) effects. In greater detail, PEs belonging to the update neighborhood centered on winner unit $PE_{w1(t)}$ are affected by a lateral excitatory signal generated by the winner, which is shaped like a "bell curve" (bubble strategy), such that [4]:

$$\alpha_i(t) = \epsilon(t) \cdot e^{-(\|r_i - r_{w1(t)}\|^2/\sigma(t)^2)} \tag{11}$$

where $\alpha_i(t)$ is the learning rate of processing unit PE_i at time t, r_i and $r_{w1(t)}$ denote the spatial coordinates of the output unit PE_i and winner unit $PE_{w1(t)}$ in the external lattice, and $\epsilon(t)$ and $\sigma(t)$ are the two user-defined monotone decreasing functions of time describing the amplitude and standard deviation of the Gaussian function respectively. Note that in Equation (11), distance $\|r_i - r_{w1(t)}\|$ is equivalent to a neighborhood-ranking of the processing units within the external lattice, i.e., SOM differs from: a) the Neural Gas (NG) algorithm, which employs the neighborhood-ranking of the reference vectors within the input space [9]; and b) other clustering algorithms (e.g., "maximum-entropy" clustering [9], FLVQ and FOSART, see Sections 10 and 13) where the adaptation step is a function of the absolute distance of the reference vectors from the current input pattern within the input space. Analysis of Equation (11) reveals that: i) when $\sigma(t) = 0$, then nodes of SOM become purely competitive and SOM becomes equivalent to a hard (crisp) c-means clustering procedure [9]; and ii) when $\epsilon(t) = 0$, then SOM reaches termination (no template vector is attracted by any input pattern). • SOM employs the Kohonen weight adaptation rule to position optimally cluster prototypes that belong to the resonance domain (update neighborhood) centered on the winner PE, such that:

$$T_i(t+1) = T_i(t) + \alpha_i(t) \cdot (X(t) - T_i(t)), \tag{12}$$

where $\alpha_i(t)$ is computed by means of Equation (11) such that if $i = w1(t)$ then $\alpha_{w1(t)}(t) = \epsilon(t)$. It is important to observe that Equation (12) is related to on-line (McQueen's) k-means [16], [33], [43], whose batch (Lloyd's or Forgy's) version [16] is a special case of the EM optimization of a Gaussian mixture [33]. Nonetheless, as long as Equation (11) features $\sigma(t) > 0$ (i.e., as long as the SOM soft learning strategy is not equivalent to a WTA strategy), one cannot specify a cost function that is minimized by Equation (12), i.e., there exists no cost function yielding Equation (12) as its gradient [9], [34], [65]. SOM instead features a set of potential functions, one for each node, to be independently minimized following a stochastic (on-line) gradient descent [65]. In [15], a cost function that leads to an update strategy that is similar to, but not precisely the same as, Equations (11) and (12) is discussed. This cost function was introduced in a nonneural context to design an optimal vector quantizer codebook for encoding data for transmission along a noisy channel [17].

9.3 Limitations

Despite its many successes in practical applications, SOM contains some major deficiencies (many of which are acknowledged in [5]).

• Since SOM does not minimize any known objective function, termination is not based on optimizing any model of the process or its data [7].
• On-line SOM is order dependent (due to on-line learning), i.e., the final weight vectors are affected by the order of the input sequence [7].
• Prototype parameter estimates may be severely affected by noise points and outliers. This is due to the fact that learning rates in SOM are computed as a function of the number of input presentations and node positions in the grid, while they are independent of the actual distance separating the input pattern from the cluster template.
• The size of the output lattice, the step size and the size of the resonance neighborhood must be varied empirically from one data set to another to achieve useful results [7].
• Probability density function (pdf) estimation is not achieved [34]. Attempts have been made to interpret the density of codebook vectors as a model of the input data distribution but with limited success [26],[27], [34], [51].
• It should not be employed in topology-preserving mapping when the dimension of the input space is larger than three. In fact, SOM tries to form a neighborhood-preserving inverse mapping ϕ_S^{-1} from lattice G to input manifold X, but not necessarily a neighborhood preserving mapping ϕ_S from X to G [49]. To obtain a topologically correct map by running the SOM algorithm, the topological (adjacency) structure of the preset graph G has to match the topological structure of the unknown manifold X [49].

9.4 Advantages

• Owing to its soft competitive implementation, SOM is expected to be less likely trapped in local minima and less likely to generate dead units than hard competitive alternatives [9], [27], [62].
• Batch SOM is order independent, i.e., the final weight vectors are not affected by the order of the input sequence.
• SOM can be employed as a vector requantization system. For some authors, "SOM was not intended for pattern classification. Rather, SOM attempts to find topological structures in the input data and display them in one or two dimensions" [54], i.e., SOM can be employed in data visualization tasks because "SOM simply attempts to achieve a consistent spatial mapping of the training vectors to (usually) two dimensions" [2]. More precisely, it can be stated that SOM can be employed in topology-preserving mapping iff the dimension of the input space is not larger than the dimension of the output lattice [49].

9.5 Architectural features

The main features of SOM are summarized in Table 2.

10 FLVQ

FLVQ [6] (which was first called FKCN [7]) has quickly gained popularity as a fairly successful batch clustering algorithm.

FLVQ design aims to improve performance and usability of Kohonen's on-line VQ and SOM algorithms by combining the on-line Kohonen weight adaptation rule (12) with the fuzzy set membership function proposed by the batch FCM algorithm [6], [7]. This allows FLVQ to compute the learning rate and the "size" of the update neighborhood directly from the data. Thus, FLVQ employs a smaller set of user defined parameters than SOM. Unlike SOM, FLVQ is a batch clustering method and employs metrical neighbors in the input space rather than topological neighbors belonging to an output lattice. In particular, to compute the adaptation step, FLVQ takes into account the absolute distances of reference vectors with respect to the current input pattern, while SOM employs the neighborhood ranking of the processing units within the external lattice.

Recent advances in the field have presented a broad family of batch FLVQ algorithms formally defined as a class of cost function minimization schemes. Hereafter, this class of batch vector quantizers will be referred to as the Extended FLVQ Family (EFLVQ-F) [55], [56], [57]. FLVQ updating can be seen as a special case of EFLVQ-F learning schemes for a restricted range of the weighting exponent. FLVQ is also related to several on-line fuzzy clustering algorithms such as the sequential Generalized LVQ (GLVQ) [58] and GLVQ Family algorithms (GLVQ-F) [59], and the class of on-line Fuzzy Algorithms for Learning Vector Quantization (FALVQ, whose proposed instances are termed FALVQ 1, FALVQ 2 and FALVQ 3) [54], [57]. Table 1 summarizes functional comparisons between these learning vector quantization algorithms. For a detailed analysis of these models, refer to [60].

Table 1:

		Batch updates	Sequential updates
Hard (crisp, purely) competitive learning		-	VQ
Soft competitive learning	Relative membership function where weighting exponent m is constant	FCM, EFLVQ-F[1]	GLVQ-F[2]
	Relative membership function where weighting exponent $m = m(e)$ is a function of training epochs	FLVQ	-
	Other membership functions with no weighting exponent m	-	GLVQ, FALVQ
	Width of the learning rate distrib. is constant	FCM, EFLVQ-F[1]	GLVQ[3], GLVQ-F[2] FALVQ
	Width of the learning rate distrib. decreases with time	FLVQ	-
Cooling schedule (learning rate decreases with time)		EFLVQ-F[4]	VQ, FALVQ, GLVQ, GLVQ-F[5]
No cooling schedule		FCM, FLVQ	-

[1] see [55], p. 252 ($m = 2$).
[2] see [59], p. 1068 ($m = 2$).
[3] extended to the entire net.
[4] see [55], p. 251.
[5] see [57], p. 33.

10.1 Input parameters

- FLVQ requires the user to define number c of natural groups to be detected.
- It requires the initial and final weighting exponent m_0 and m_f, controlling the "amount of fuzziness" of the algorithm. In [6], the heuristic constraint $7 > m_0 > m_f > 1.1$ is recommended.
- Parameter e_{max} is defined as the maximum number of epochs.
- A convergence error ϵ is employed in the termination strategy.

10.2 Description of the algorithm

• Descending FLVQ employs a weighting exponent $m = m(e)$ monotonically decreasing with processing time e, where e is the number of processing epochs. The decreasing expression of $m(e)$ is [1]:

$$m(e) = m_0 - (e \cdot \triangle m) \qquad \text{where } \triangle m = (m_0 - m_f)/e_{max} \tag{13}$$

• FLVQ employs metrical neighbors in the input space (analogously to GLVQ, GLVQ-F and FALVQ). In particular, FLVQ computes a membership function providing the degree of compatibility of an input pattern with the vague concept represented by a cluster center. To compute this inter-pattern similarity measure, FLVQ combines winner and non-winner information as follows:

$$u_{i,k}(e) = \frac{\left(\frac{1}{d(X_k(e),T_i(e))^2}\right)^{\frac{1}{(m(e)-1)}}}{\sum_{j=1}^{c}\left(\frac{1}{d(X_k(e),T_j(e))^2}\right)^{\frac{1}{(m(e)-1)}}}, \qquad i = 1, 2, ..., c, \;\; k = 1, 2, ..., n, \tag{14}$$

where c is the total number of categories, n is the total number of input patterns, $X_k(e)$ is an input pattern and $T_i(e)$ is a cluster template at epoch time e, and $d(X_k(e), T_i(e))$ is the Euclidean inter-pattern distance between $X_k(e)$ and $T_i(e)$. Equation (14) is a relative or probabilistic membership function because it satisfies the three conditions required to state that c fuzzy subsets are a fuzzy c-partition of the input space (see Section 2.1).

• The FLVQ learning rule is:

$$T_i(e+1) = \quad T_i(e) + \eta_i(e) \cdot \sum_{k=1}^{n} w_{i,k}(e)(X_k(e) - T_i(e))$$

$$= T_i(e) + \sum_{k=1}^{n} \alpha_{i,k}(e)(X_k(e) - T_i(e)) = \sum_{k=1}^{n} \alpha_{i,k}(e) \cdot X_k(e), \quad i = 1, 2, ..., c, \tag{15}$$

where

$$w_{i,k}(e) = (u_{i,k}(e))^{m(e)}, \tag{16}$$

$$\eta_i(e) = \frac{1}{\sum_{k=1}^{n} w_{i,k}(e)}, \tag{17}$$

$$\alpha_{i,k}(e) = \eta_i(e) \cdot w_{i,k}(e). \tag{18}$$

It can be observed that if $d(X_k(e), T_i(e)) \to 0$, $i = 1, ..., c$, then $u_{i,k}(e) \to 1/c$. This causes "the relative membership problem of FCM" [61]. It means that since Equation (14) provides membership values that are relative numbers, then noise points and outliers may have significantly high membership values and may severely affect the prototype parameter estimate (15). Note that if all input patterns are outliers for the given set of template vectors, then $\alpha_{i,k}(e) \to 1/n$, $i = 1, ..., c$, $k = 1, ..., n$, i.e., according to Equation (15) all templates collapse into the center of gravity of the input data set. It can also be observed that Equation (15) belongs to the same class of weight adaptation rules employed in the batch form of the SOM algorithm and in the batch and statistically firm EM optimization of Gaussian centers in a mixture model [5], [33], [34]. Equation (15) shows that when

$m(e) = m$ is fixed, then FLVQ is equivalent to FCM [6]. It can be proved that by decreasing its weighting exponent $m(e)$, descending FLVQ tends to reduce the width of the learning rate distribution by means of model transitions. In this case, descending FLVQ satisfies Kohonen's second learning constraint requiring the size of the update neighborhood to decrease monotonically (see Section 9). Unfortunately, due to improper initialization of parameter $m(e)$, the initial learning phase of FLVQ may lead the algorithm to perform trivial (non-adaptive) vector quantization. This is demonstrated by the following analysis of FLVQ asymptotic behaviors.
Asymptotic case A: $m(e) \to \infty$ causes trivial vector quantization. It is easy to verify that [60]

$$\lim_{m(e)\to\infty} T_i(e+1) = T_i(e) + \frac{1}{n}\sum_{k=1}^{n}(X_k(e) - T_i(e)) = \frac{1}{n}\sum_{k=1}^{n} X_k(e), \qquad i = 1, 2, ..., c. \quad (19)$$

Equation (19) shows that when $m(e) \to \infty$, all input patterns are weighted equally whatever category i may be. Therefore, all FLVQ cluster centers collapse into the same point, i.e., all input patterns are mapped into the same prototype which is the center of gravity of the input data set. FLVQ shares with FCM this asymptotic behavior leading to trivial vector quantization [60]. Unfortunately, Equations (14) to (18) applied iteratively are unable to make identical centroids move away from each other, i.e., whenever centroids converge to the center of gravity (grand mean) of the input data set then separate processing units can no longer be adapted to different degrees. This asymptotic behavior is acknowledged in [1], where it is recommended to keep $m(e)$ away from infinity to prevent numerical instability of FLVQ. Our analysis suggests that rather than numerical instability, limiting condition $m(e) \to \infty$ causes prototype collapse in both FLVQ and FCM.
Asymptotic case B: $m(e) \to 1^+$ produces hard competitive learning plus trivial dead unit relocation. It can be demonstrated that $\lim_{m(e)\to 1^+} w_{i,k}(e) = 1$ if neuron i is the winner unit for pattern X_k, otherwise $w_{i,k}(e)$ tends to zero. As a consequence,

$$\lim_{m(e)\to 1^+} \alpha_{i,k}(e) = \begin{cases} 1/n_i, \text{ if neuron } i \text{ is the winner unit for } X_k(e), \\ \quad \text{where } 1 \le n_i \le n \text{ is the total number of times} \\ \quad \text{neuron } i \text{ is the best-matching} \\ \quad \text{unit during the current epoch } t; \\ 0, \text{ if neuron } i \text{ is not the winner unit for } X_k(e), \\ \quad \text{and if } n_i \ge 1, \text{ i.e., unit } i \text{ is the winner for} \\ \quad \text{at least one pattern } X_h(e) \ne X_k(e),\ h \in \{1, n\}; \\ 1/n, \text{ if } n_i = 0, \text{ i.e., if neuron } i \text{ is not the winner unit either for } X_k(e) \\ \quad \text{or for any other input pattern, i.e., if neuron } i \text{ is a dead unit;} \end{cases} \quad (20)$$

Therefore,

$$\lim_{m(e)\to 1^+} T_i(e+1) = \begin{cases} \frac{\sum_{X_k(e)\in\{X_i(e)\}} X_k(e)}{n_i}, \text{ if } n_i \ge 1; \\ \frac{\sum_{k=1}^{n} X_k(e)}{n}, \text{ if } n_i = 0, \end{cases} \quad (21)$$

where set $\{X_i(e)\}$ consists of input patterns featuring $T_i(e)$ as their best-matching template at epoch number e. Equation (21) shows that if $m(e) \to 1^+$, then both FCM and FLVQ become similar but not equivalent to a batch (Forgy's) c-Means algorithm [16]. Forgy's

c-Means features a singularity condition when its hard competitive prototype updating, $\sum_{X_k(e) \in \{X_i(e)\}} X_k(e)/n_i$, deals with a dead unit ($n_i = 0$). In this case, the prototype update cannot be calculated. On the contrary, when Equation (21) encounters dead units, these cluster centers are moved to the center of gravity of the input data set. In the case of a descending FLVQ implementation, since the non-recoverable collapse of dead units described by Equation (21) eventually occurs at the end of the iteration process (when $m(e) \to 1^+$), then this loss of resources cannot be considered as an asymptotic deficiency of the algorithm.

Model case C: $m(e) \in (1, \infty)$. When descending weighting exponent $m(e)$ is kept away from infinity (asymptotic case A) and from value 1 (asymptotic case B), the update neighborhood includes all c nodes characterized by unequal excitation.

To summarize, if **asymptotic case A** is avoided in descending FLVQ, then, on the one hand, the width of the learning rate distribution decreases with time as **model case C** approaches **asymptotic case B**, i.e., FLVQ behaves consistently with the second Kohonen constraint. These theoretical conclusions about transitions of the FLVQ learning strategy as a function of decreasing $m(e)$ (increasing epoch time e) are consistent with those regarding FCM, EFLVQ-F and GLVQ-F [55], [56], [57], [59], all systems employing Equation (14) as their relative membership function (see Table 1). This analysis is also consistent with the heuristic choice $7 > m_0 > m_f > 1.1$ recommended in [6]. On the other hand, learning rate values do not necessarily decrease with time, i.e., FLVQ does not behave consistently with the first Kohonen constraint, its update mechanism being roughly opposite to that of other clustering algorithms, like SOM. For example, in SOM, all c learning rates (winner as well as non-winners) decrease towards 0 as t increases; vice versa, in FLVQ, asymptotic Equations (19) and (20) show that learning rate $\alpha_{i,k}(e)$ may increase up to $1/n_i$ for the winner neuron when e increases (starting at $e = 1$ from a value which may be close to $1/n$, see **asymptotic case A**), while the other $(c - 1)$ rates tend towards zero or $1/n$ (see **asymptotic case B**). Intuitively, this learning policy is not desirable because it does not provide prototypes with the large initial plasticity values required to pursue fast (rough) initial learning.

• In [6], it is clarified that FLVQ, like SOM, does not optimize any known objective function, and that it is expected to reach termination when the FCM objective function is approximately minimized [7]. In [55], [56], [57], EFLVQ-F learning schemes are formally derived to minimize a given functional when m is constant. It is also shown that FLVQ updating can be seen as a special case of EFLVQ-F learning schemes for a restricted range of the weighting exponent. This does not mean, however, that FLVQ minimizes the EFLVQ-F functional since the hypothesis $m = \textit{constant}$ does not hold true for FLVQ. We conclude that despite recent advances in the field, the objective function minimized by FLVQ is still unknown, just as the one minimized by SOM is unknown [60].

10.3 Limitations

• Since FLVQ does not minimize any known objective function, termination is not based on optimizing any model of the process or its data [7].

• FLVQ is affected by the relative membership problem (high sensitivity to noise, i.e., low robustness [62]).

• It does not provide prototypes with large initial plasticity values required to pursue fast (rough) initial learning.
• FLVQ features instability when its traditional termination criterion is employed, such that if $\sum_{i=1}^{c} d(T_i(e), T_i(e-1))^2 \leq \epsilon$, then FLVQ is terminated. Experimental tests reveal that clustering results improve when convergence is reached at m_f values close to 1, while termination parameter ϵ is ignored (see Section 10.1) [60].
• It does not provide pdf estimation.
• It cannot be employed in topology-preserving mapping.

10.4 Advantages

• Owing to its soft competitive implementation, FLVQ is expected to be less likely trapped in local minima and less likely to generate dead units than hard competitive alternatives (e.g., FCM) [9], [27], [62].
• In FLVQ, due to batch learning, the final weight vectors are not affected by the order of the input sequence when its traditional termination criterion is removed.
• With respect to SOM, FLVQ requires a smaller set of input parameters (its learning rate and the size of the update neighborhood being computed directly from the data).
• FLVQ can be employed as a vector requantization system.

10.5 Architectural features

The main features of FLVQ are summarized in Table 2.

11 Fuzzy ART

In recent years, several ART-based models have been presented. ART 1 categorizes binary patterns but features sensitivity to the order of presentation of the random sequence [21]. This finding led to the development of the Improved ART 1 system (IART 1), which is less dependent than ART 1 on the order of presentation of the input sequence [69]. The Adaptive Hamming Net (AHN), which is functionally equivalent to ART 1, optimizes ART 1 both in terms of computation time and storage requirement [23]. ART 2, designed to detect regularities in analog random sequences, employs a computationally expensive architecture which presents difficulties in parameter selection [22]. To overcome these difficulties, the Fuzzy ART system was developed as a generalization of ART 1 [19], [20]. This means, however, that ART 1-based structural problems may also affect Fuzzy ART.

The structured organization of ART systems is made up of two subsystems, termed attentional and orienting subsystem. Although in its original form the ART 1 attentional subsystem employs bottom-up (feed-forward) and top-down (feed-backward) connections, it is easy to prove that this module is mathematically equivalent to an attentional subsystem where feed-forward connections are adopted exclusively [24]. For example, AHN is a feed-forward network functionally equivalent to ART 1 [23]. This simplification yields, as a major consequence, a change in the meaning of the term "resonance" as traditionally applied to ART 1-based systems. This term should no longer indicate "the basic feature of all ART systems, notably, pattern-matching between bottom-up input and top-down learned

prototype vectors" ([19], p. 760), just as the term "resonance" has never been applied to pattern matching activities performed by a feed-forward network, e.g., SOM. In our view, the term "resonance", as employed in ART, means rather that all ART 1-based algorithms share the same modular architecture, consisting of:
i) a completely generic (unsupervised), flat (without hidden layers), feed-forward (bottom-up) pattern recognition network (as Kohonen's networks, e.g., VQ and SOM), termed an attentional subsystem; and
ii) a supervised/unsupervised knowledge interface unit, termed an orienting subsystem, where the quality of unsupervised bottom-up pattern recognition is compared to top-down requirements (expectations, or prior knowledge) provided by the external environment (supervisor). In the orienting subsystem, if unsupervised knowledge matches external expectations, then "resonance" occurs. This means that the unsupervised pattern recognition activity of the attentional module is reinforced according to a reinforcement learning mechanism, i.e., prototype adaptation takes place. If resonance does not occur, then the orienting subsystem allows the attentional module to increase its resources (processing elements) to meet external requirements.

Fuzzy ART requires a preprocessing stage where either input pattern normalization or complement coding is used to prevent category proliferation. Input data normalization loses vector-length information. Complement coding normalizes input vectors while preserving their amplitude information, but it doubles the number of network connections [19], [20].

It can be proved that ART 1-based systems are all affected by the same design inconsistency: they all employ an inherently asymmetrical architecture to perform an inherently symmetrical task - the assessment of an interpattern degree of match [24].

11.1 Input parameters

To simplify the discussion, our approach is to deal with a finite training data set $\{X\}$, consisting of n input patterns X_k, $k = 1, ..., n$, where $X_k \in \mathcal{R}^D$, such that D is the dimensionality of the input space. The input data set is repeatedly presented to the network until a termination criterion is satisfied.

• Fuzzy ART employs parameter $\alpha > 0$ (e.g., $\alpha \in [0.001, 1)$, [20]) to break ties, i.e., to bias the function in favor of the longer of two template vectors (see Equation (22)). A typical α value is 0.001 [20].
• It requires vigilance threshold $\rho \in [0, 1]$ as a relative number representing external expectations. In detail, ρ controls neuron proliferation (see Equation (23)), such that coarser grouping of input patterns is obtained when the vigilance parameter is lowered. Parameters ρ and α are interrelated as illustrated in [70] (when ρ decreases, α must also decrease).
• Learning rate β is independent of time and set within range $(0, 1)$ (see Equation (24)).
• It requires a termination threshold, e.g., the maximum number of epochs e_{max}, i.e., the number of times the finite training data set is repeatedly presented to the network.

11.2 Description of the algorithm

• The Fuzzy ART attentional subsystem computes activation values as a set of relative numbers. Each activation value represents the "unidirectional" (asymmetrical) degree to which the input pattern matches a template vector. In other words, the Fuzzy ART *activation function* is asymmetric with respect to the input and template vector pair, i.e., it provides no assessment of the degree to which the template vector matches the input pattern. The best-matching unit is selected as the one featuring the largest activation value. The Fuzzy ART orienting subsystem overlooks the pattern recognition activity performed by the attentional subsytem by computing the value of a "unidirectional" (asymmetrical) *match function*. This match value is a relative number representing the degree to which the winner template matches the input pattern. Next, *vigilance testing* is performed, where the match value is compared to the user-defined vigilance threshold ρ. If the match value is above the vigilance threshold, then the winner template is updated, otherwise a *mismatch reset condition* and a *search process* are run in sequence. Because the winner template detected by the activation function is not necessarily the best-matching template according to the match function, then the orienting subsystem inhibits the winner neuron and submits the processing element featuring the second largest activation value to the vigilance test. The sequence of operations, consisting of one mismatch reset condition and a search process, is repeated until either the vigilance test is passed or no more nodes are available for testing.

• In the attentional subsystem, for a given pattern $X(t) = X_k(t), \forall k \in \{1, n\}$, where n is the finite size of the training set and t is the number of input pattern presentations, winner unit $PE_{w1(t)}$ is detected as the neuron that satisfies condition $\mu_{w1(t)}(t) = \max\{\mu_i(t), i = 1, ..., c\}$, where activation value $\mu_i(t)$ is computed as:

$$\mu_i(t) = AF(T_i(t), X_k(t)) = \frac{\sum_{d=1}^{D} \min\{T_i^d(t), X_k^d(t)\}}{\alpha + \sum_{d=1}^{D} T_i^d(t)}, \qquad i = 1, ..., c, \quad \forall k \in \{1, n\}, \quad (22)$$

where $AF(\cdot)$ is the activation function, c is the total number of neurons, D is the dimensionality of the input space, $T_i^d(t)$ is the d-th component of vector $T_i(t)$ and α is the user-defined parameter (see Section 11.1). Equation (22) is an inter-pattern similarity measure that satisfies requirement (iv) presented in Section 2.1, i.e., it can be considered an absolute fuzzy membership function. Equation (22) measures to what normalized degree pattern $X_k(t)$ matches template $T_i(t)$ but it does not assess the reverse situation, i.e., to what degree $T_i(t)$ matches $X_k(t)$. In other words, this activation function is not symmetrical with respect to $X_k(t)$ and $T_i(t)$, i.e., $AF(T_i(t), X_k(t)) \neq AF(X_k(t), T_i(t))$.

• In the orienting subsystem, Fuzzy ART employs a vigilance test defined as follows:

$$MF(T_{w1(t)}(t), X_k(t)) = \frac{\sum_{d=1}^{D} \min\{T_{w1(t)}^d(t), X_k^d(t)\}}{\sum_{d=1}^{D} X_k^d(t)} \geq \rho, \quad (23)$$

where $MF(\cdot)$ is the match function that measures to what normalized degree template $T_{w1(t)}(t)$ matches pattern $X_k(t)$ while it does not assess the reverse situation, i.e., $MF(T_{w1(t)}(t), X_k(t)) \neq MF(X_k(t), T_{w1(t)}(t))$. In line with $AF(\cdot)$, $MF(\cdot)$ is an inter-pattern similarity measure that satisfies requirement (iv) presented in Section 2.1, i.e., it can be considered an absolute fuzzy membership function. If the vigilance test is satisfied, then the attentional subsystem is activated to sequentially adjust the winner template according to Equation (24). Otherwise, the mismatch reset condition and search process are activated (see above).

• Fuzzy ART employs a WTA strategy. The weight transformation law applied to the winner template is [19]:

$$T_{w1(t)}(t+1) = (1-\beta) \cdot T_{w1(t)}(t) + \beta \cdot (X_k(t) - T_{w1(t)}(t)). \tag{24}$$

Note that unlike Kohonen's clustering networks (see Section 9), Fuzzy ART employs no cooling schedule because the learning rate is constant in time.
• When no existing neuron satisfies the vigilance test, then $c = c+1$, a new processing unit PE_c is allocated by the orienting subsystem and the new template vector is initialized as $T_c(t+1) = X_k(t)$.
• With regard to Equations (22) and (23), Fuzzy ART substitutes the operators employed in the ART 1-based activation function and match function with fuzzy-like operations (intersection and cardinality). As observed by Simpson [8], to be correctly interpreted as fuzzy operations, these operations would have to be applied to fuzzy set membership values, rather than to the parameters (pattern and template vectors) of absolute fuzzy set membership functions.[3] To summarize, Fuzzy ART employs two absolute fuzzy membership functions, i.e., it employs fuzzy set-theoretic concepts. Nonetheless, since it applies no soft competitive learning strategy, Fuzzy ART cannot be termed fuzzy as defined in Section 7.

11.3 Limitations

• Since Fuzzy ART does not minimize any known objective function, its termination is not based on optimizing any model of the process or its data [7].
• Owing to its hard competitive implementation, Fuzzy ART is more likely to be trapped in local minima and to generate dead units than soft competitive alternatives [9], [27], [62].
• Fuzzy ART is order dependent due to on-line learning and example-driven neuron generation. Experimental evidence [69], as well as theoretical analysis [24], reveal that Fuzzy ART sensitivity to the order of presentation of the input sequence is also due to some inconsistencies detected in the system's implementation [24], [69].
• Fuzzy ART may be severely affected by noise points and outliers, i.e., it may fit the noise and not just the data. In other words, Fuzzy ART may be affected by overfitting, because a single poorly mapped pattern suffices to initiate the creation of a new unit, while no noise category removal mechanism is employed by the system.
• Since its learning rate is independent of time, Fuzzy ART lacks stability because of excessive plasticity, i.e., the processing of a new data set may affect (i.e., move through the input space) templates detected by the system during the previous learning phase. Nonetheless, since the length of every template can only decrease with learning, templates cannot "cycle", i.e., Fuzzy ART does have a sort of stability [24].
• Fuzzy ART is time-consuming with respect to functionally equivalent implementations [24]. In its traditional implementation consisting of c nodes (see [19]), for each input pattern the Fuzzy ART search process requires a minimum of one up to c iterations to detect the winner unit that features the largest activation value among the units capable of satisfying the vigilance test. Several ART-1 based models, either functionally equivalent to Fuzzy ART (e.g, AHN [23]) or not functionally equivalent to Fuzzy ART (e.g., the Fuzzy SART model [24]), reduce the number of searches to one , whatever the input pattern may be.

[3] It is curious to observe that the same criticism can be applied to the Fuzzy Min-Max clustering algorithm presented by Simpson [8]

• It requires input data preprocessing (e.g., normalization or complement coding; in the first case data vector-length information is lost, while in the second case the number of network connections doubles) to prevent category proliferation [19].
• It does not provide pdf estimation.
• It cannot be employed in topology preserving mapping.

11.4 Advantages

• Feed-back interaction between attentional and orienting subsystems allows Fuzzy ART to self-adjust its size depending on the complexity of the clustering task.
• Distinct sample vectors are employed to initialize reference vectors. This choice reduces the risk of dead unit formation and may reduce computation time with respect to random initialization.
• Fuzzy ART can be employed as a vector requantization system.

11.5 Architectural features

The main features of Fuzzy ART are summarized in Table 2.

12 GNG

GNG combines the growth mechanism inherited from the earlier proposed Growing Cell Structures [73] with the synapse geration rule CHR [49]. GNG is capable of generating and removing both synaptic links and PEs, i.e, GNG belongs to the class of FSONN models (see Section 5). In particular, starting with very few units (generally, two), one new unit is inserted every λ adaptation steps near the unit featuring the largest local error measurement. In other words, GNG employs a mini-batch approach (see Section 3) to decide where to locate new PEs: by accumulating error information over a number of λ pattern presentations, i.e., by averaging over the noise on the data, GNG does not allow any single poorly mapped pattern to initiate the creation of a new unit. In [75], it is anticipated that the future development of GNG will employ, besides the neuron insertion criterion described above, the following rule for neuron removal: every λ adaptation steps, the unit featuring lowest utility for error reduction is removed. This utility measure is defined as the increase in overall distorsion error occuring if the unit of interest were removed from the set of templates. The utility $U(T_{w1(t)}(t))$ of template vector $T_{w1(t)}(t)$, $w1(t) \in \{1, c\}$, where c is the total number of neurons and t is the number of input pattern presentations is defined as follows [76]:

$$U(T_{w1(t)}(t)) = \sum_{X(t^*)\in\{M_{w1(t)}\}} d(X(t^*), T_{w2(t^*)}(t^*)) - d(X(t^*), T_{w1(t^*)}(t^*)), \quad t^* \leq t, \qquad (25)$$

where $d(X(t^*), T_{w1(t^*)}(t^*))$ is the Euclidean inter-pattern distance between vectors $X(t^*)$ and $T_{w1(t^*)}(t^*)$ at presentation time t^*, $T_{w2(t^*)}(t^*)$ is the second best matching template for vector $X(t^*)$, such that $w2(t^*) \neq w1(t^*)$, and $\{M_{w1(t)}\}$ is the subset of the input pattern presentation sequence featuring template of unit $PE_{w1(t)}$ as the closest reference vector, i.e., $\{M_{w1(t)}\} = \{X(t^*)$: unit $PE_{w1(t)}$ satisfies the condition $d(X(t^*), T_{w1(t)}(t^*)) \leq d(X(t^*), T_i(t^*)), i = 1, ..., c,\ t^* \leq t\ \}$.

An additional mini-batch learning policy is employed to remove synaptic links whose utility has progressively faded away while the presentation of the input sequence goes on [27]. To summarize, GNG exploits mini-batch learning techniques for inserting processing units, deleting processing units and deleting synapses, and an example-driven CHR to generate synaptic links, the latter strategy being more sensitive to the presence of noise than mini-batch learning.

12.1 Input parameters

To simplify the discussion, our approach is to deal with a finite training data set $\{X\}$, consisting of n input patterns X_k, $k = 1, ..., n$, where $X_k \in \mathcal{R}^D$, such that D is the dimensionality of the input space. The input data set is repeatedly presented to the network until a termination criterion is satisfied.

• GNG employs two user-defined learning rates: $\epsilon_1 \in (0, 1)$ is applied to the winner neuron, while $\epsilon_n \in (0, 1)$, such that $\epsilon_n < \epsilon_1$ is applied to neurons belonging to the update neighborhood. Typical values are: $\epsilon_1 = 0.05$, $\epsilon_n = 0.0006$ [27].
• Parameter $\lambda \in \mathcal{N}^+$ controls neuron generation at adaptation steps (see above).
• Parameters α and β are used to decrease the error variables, so that more recent signals are weighted more heavily than previous ones [73]. Typical values are: $\alpha = 0.5$, $\beta = 0.0005$ [27].
• Parameter $a_{max} \in \mathcal{N}^+$ is the maximum age of a synaptic link. A typical value is: $a_{max} = 88$ [27].
• GNG employs a termination parameter, such as the maximum number of neurons c_{max} [27].

12.2 Description of the algorithm

• In line with SOM, GNG employs geometrical (distance) relationships in the input (measurement) space to enforce competitive learning mechanisms. Let us consider: a) winner unit $PE_{w1(t)}$ as the one featuring the shortest Euclidean inter-pattern distance $d(X_k(t), T_{w1(t)}(t)) \leq d(X_k(t), T_i(t))$, $i = 1, ..., c$, where t is the number of input pattern presentations; and b) the second best unit $PE_{w2(t)}$ as the one featuring activation value $d(X_k(t), T_{w2(t)}(t)) \leq d(X_k(t), T_i(t))$, $i = 1, ..., c$, $i \neq w1(t)$. Exploitation of the Euclidean inter-pattern distance in competitive learning shapes neuron receptive fields as Voronoi polyhedra (see Section 5).
• As soon as neuron $w1(t)$ is detected, the local error variable of winner neuron $w1(t)$ is updated; for example, for requantization tasks, the increase of accumulated error is defined as

$$E_{w1(t)}(t) = E_{w1(t)}(t-1) + (d(X_k(t), T_{w1(t)}(t))^2, \tag{26}$$

otherwise, for probability density function estimation,

$$E_{w1(t)}(t) = E_{w1(t)}(t-1) + 1. \tag{27}$$

• In line with SOM, topological relationships among PEs belonging to an output lattice are employed to detect soft competitive (competitive and cooperative) effects. In particular, GNG applies an update equation to the winner unit $PE_w1(t)$ and to its topologically adjacent neighbors, i.e., the resonance neighborhood consists of all PEs directly connected to the winner (such that one synaptic link, identified as $S_{w1(t),i}$, $i \in \{1, c\}$, $i \neq w1(t)$, exists at time t). Therefore, due to dynamic generation/removal of PEs and synaptic links, the resonance domain changes with time to include PEs that are topologically adjacent to winner $w1(t)$. Nonetheless, this behavior does not strictly satisfy the Kohonen constraint requiring the size of the update neighborhood to decrease monotonically with time.

• Templates belonging to the update neighborhood are adapted in line with the Kohonen weight adaptation rule (see Equation (12)). The update equation is

$$T_i(t+1) = T_i(t) + \epsilon_i(t) \cdot (X_k(t) - T_i(t)), \tag{28}$$

where $\epsilon_i(t) = \epsilon_1$, which is fixed by the user (see Section 12.1), if $i = w1(t)$, and $\epsilon_i(t) = \epsilon_n$ and if vertice PE_i is adjacent to the winner at time t (see above).

• With regard to generation of PEs, GNG employs the following strategy: one new units is inserted every λ adaptation steps near the unit featuring the largest local error measure, identified as unit q. In particular, the receptive field center of the new unit is located half way between the templates of units q and f, where f identifies the neuron featuring the maximum accumulated error among the neighbors of q (i.e., synaptic link $S_{q,f}$ does exist). The local error of the new unit is set to $(E_q + E_f)/2$; then E_q and E_f are decreased by a fraction α.

• With regard to generation of synaptic links, GNG employs the following strategy. As soon as units $PE_{w1(t)}$ and $PE_{w2(t)}$ are detected, CHR is applied to $PE_{w1(t)}$ and $PE_{w2(t)}$ so that if connection $S_{w1(t),w2(t)}$ does not exist, it is generated (see Section 5). In either case, the age of the connection between $PE_{w1(t)}$ and $PE_{w2(t)}$ is reset to zero (the connection is "refreshed").

• With regard to removal of superfluous synaptic links and neurons, GNG employs the following strategy. After CHR is applied to the unit pair $PE_{w1(t)}$ and $PE_{w2(t)}$, and connection $S_{w1(t),w2(t)}$ is refreshed, the age of all edges emanating from $PE_{w1(t)}$ is increased by one. Among these connections, edges whose age is larger than a_{max} are removed. If this results in units having no emanating edge, such units are removed.

• At the end of each input pattern presentation: i) the error variables of all units are decreased by a fraction β; and ii) the termination test is checked (possibly involving parameter c_{max} or the mean accumulated error).

12.3 Limitations

• Since GNG does not minimize any known objective function, termination is not based on optimizing any model of the process or its data [7].

• Since its neurons feature no "cooling schedule" (i.e., the learning rate does not satisfy Kohonen's first learning constraint), GNG lacks stability because of excessive plasticity, i.e., the processing of a new data set can radically alter maps detected by the system during the previous processing stage (try simulations on the web at the address reported in [27]).

• It is not very easy to use, since it requires seven user-defined parameters whose meaning is not always straightforward.

• Sample vectors are not employed to initialize reference vectors. In particular, new neurons can be initialized outside the input manifold, and this increases computation time before

termination is reached.
• It combines mini-batch learning techniques (for neuron generation and removal, and for synapse removal) with example-driven generation of synapses.

12.4 Advantages

• Owing to its soft competitive implementation, GNG is less likely to be trapped in local minima and to generate dead units than hard competitive alternatives [9], [27], [62]. Note that although GNG employs a soft competitive strategy, its update neighborhood size does not decrease with time, as required by Kohonen's constraints.
• Due to its insertion strategy, based on accumulated errors, it is robust against noise, and this avoids overfitting.
• In the tests provided involving the 2-D two-spiral data set (194 patterns, 2 classes), a two-stage hybrid classifier (supervised + unsupervised) exploiting GNG as its first layer performs better classification than other distributed systems found in the literature [73]. The excellent adaptivity of GNG can be directly tested by simulations on the web [27].
• GNG is computationally efficient because its computation time increases linearly as ($c+$ No. of links).
• GNG can be employed in: a) vector quantization, b) density function estimation, and c) structure detection in input data to be mapped in a topologically correct way onto submaps of an output lattice pursuing dimensionality reduction.

12.5 Architectural features

The main features of GNG are summarized in Table 2.

13 FOSART

Created to overcome the GNG deficiencies listed in Section 12.3, FOSART is an on-line learning algorithm which combines properties of SOM [5], GNG [27], FCM [6], and Fuzzy SART [24] algorithms. FOSART employs a relative fuzzy membership function, derived from the one employed by FCM and Fuzzy SART, to compute activation values, and a Gaussian Basis Function (GBF) to compute absolute (possibilistic) fuzzy membership values (see Section 3). Generalizing the soft competitive learning strategy adopted by GNG, FOSART considers the whole output map to which the best-matching unit belongs as the resonance neighborhood of the best-matching unit $PE_{w1(t)}$. In line with Fuzzy ART, FOSART applies an ART-based, example-driven vigilance test to control neuron generation [19], [24]. Similarly to GNG, FOSART employs a new version of CHR, termed Constrained Competitive Hebbian Rule (CCHR) to generate synaptic links on an example-driven basis [49]. In line with GNG, FOSART removes synaptic links as well as neurons according to a mini-batch parameter adaptation scheme. To summarize, FOSART as well as GNG belongs to the class of FSONN models (see Section 5).

13.1 Input parameters

To simplify the discussion, our approach is to deal with a finite training data set $\{X\}$, consisting of n input patterns X_k, $k = 1, ..., n$, where $X_k \in \mathcal{R}^D$, and where D is the dimensionality of the input space. The input data set is repeatedly presented to the network until a termination criterion is satisfied. Each presentation sequence is termed a training epoch.

• FOSART requires the user to define an ART-based vigilance threshold as a relative number $\rho \in (0, 1]$. Since coarser grouping of input patterns is obtained when the vigilance parameter is lowered, in biological terms vigilance threshold ρ can be interpreted as the quantity of nutrients provided by the external environment (supervisor) to the learning system for neuron proliferation.
• It employs a synapse max-to-min length ratio threshold $lr \geq 1$.
• To reach termination, FOSART requires a lower limit for the number of training epochs each node has to survive, $e_{min} \geq 1$, this parameter affecting the overall number of training epochs required by the algorithm to reach termination (consider that, in FOSART, units are generated and removed dynamically as the number of input pattern presentations, t, increases).

13.2 Description of the algorithm

Since FOSART has never been extensively presented in the literature [28], [29], and has been subjected to continuous refinements, its current version is provided hereafter.

Step 0. Initialization. Pattern counter t is set to 0. Two input patterns, X_1 and X_2, are randomly chosen from the input set and two Processing Elements (PEs), PE_1 and PE_2, are generated. The PE counter, c, is set to 2. Template vectors $T_1(t)$ and $T_2(t)$ (representing centers of receptive fields equivalent to Voronoi polyhedra in the input space) are set equal to pattern X_1 and X_2 respectively. PE-based (local) epoch counters e_1 and e_2 are initialized to 0. FOSART employs PE-based time counters to compute PE-based plasticities (learning rates). In FOSART, the "age" (local time) of a processing unit is an integer value equal to the number of times the finite input data set has been iteratively presented to the system while that processing unit exists. PE-based (local) best-matching counters, bm_1 and bm_2, are initialized to 0.

Step 1. Input pattern presentation. The pattern counter is increased by one as $t = t + 1$, and a new pattern $X_k(t)$, $\forall k \in \{1, n\}$, is presented to the network.

Step 2. Detection of processing units eligible for being resonant. Determine the best-matching unit $PE_{w1(t)}$, and the second-best unit $PE_{w2(t)}$, $w1(t)$ and $w2(t) \in \{1, c\}$, $w1(t) \neq w2(t)$, as the pair of PEs featuring, respectively, the largest and the second-largest Gaussian Radial Basis Function (GRBF) value, i.e.,

$$w1(t) = \arg \max_{i=1,...,c} \{Gaussian_i(t)\} = \arg \min_{i=1,...,c} \{d_i(t)\}, \tag{29}$$

$$w2(t) = \arg \max_{i=1,...,c; i \neq w1(t)} \{Gaussian_i(t)\} = \arg \min_{i=1,...,c; i \neq w1(t)} \{d_i(t)\}, \tag{30}$$

where GRBF is

$$Gaussian_i(t) = e^{-\frac{(d_i(t))^2}{2\sigma^2}} \in (0, 1], \tag{31}$$

such that

$$d_i(t) = d(X_k(t), T_i(t))$$

is the Euclidean distance between input pattern $X_k(t)$ and prototype vector (receptive field center) $T_i(t)$ of processing unit PE_i at presentation time t. In Equation (31), spread (resolution) parameter σ is computed as

$$\sigma = (1/\rho). \tag{32}$$

Equation (32) considers σ as being inversely related to user-defined parameter ρ; intuitively, this relationship means that when the number of PEs employed to map any given input manifold X onto an output lattice (network) G increases (this number being proportional to the ART-based severity threshold), i.e., in input space, the size of the receptive fields of neural units decreases. Note that since $Gaussian_i(t)$, $i = 1, ..., c$ feature the same spread parameter σ, then enforcing competitive learning among these GRBFs is equivalent to considering neuron receptive fields as Voronoi polyhedra [27].

Step 3. Resonance domain detection. From among processing units candidated by the attentional subsystem as being resonant the orienting subsystem selects those that match external requirements. Only these units are said to belong to the resonance domain. To select these units, the orienting subsystem employs an ART-based vigilance constraint. The vigilance test applied to the pattern matching activity of the attentional module is

$$Gaussian_{w1(t)}(t) \geq \rho, \quad \rho \in (0, 1], \tag{33}$$

where user-defined vigilance parameter ρ provides a model of top-down external requirements (expectations, or prior knowledge) provided by the external environment (supervisor). In the orienting subsystem, if unsupervised knowledge matches external expectations, then "resonance" occurs. This means that the unsupervised pattern recognition activity of the attentional module is reinforced according to a reinforcement learning mechanism, i.e., prototype adaptation takes place. If resonance does not occur, the orienting subsystem allows the attentional module to increase its resources (processing elements) to match external requirements.

Step 4(a). Resonance condition: reinforcement learning. If the vigilance test is satisfied and "resonance" occurs, the attentional subsystem is allowed to reinforce its pattern-matching activity by adjusting template vectors of units belonging to the resonance domain. The following sequence of operations is performed.

1. The best-matching counter of unit $PE_{w1(t)}$ is increased by one, i.e., $bm_{w1(t)}(t) = bm_{w1(t)}(t-1) + 1$.

2. A Constrained Competitive Hebbian Rule (CCHR), which introduces a competitive mechanism among synaptic links, is applied. CCHR requires that the ratio between

the length (computed in the input space, see below) of the longest and shortest connection emanating from any processing unit at any time to be in all cases $\leq lr$, where lr is a user-defined threshold (see Section 13.1). At this processing step, between the best and second-best matching unit, $PE_{w1(t)}$ and $PE_{w2(t)}$ respectively, CCHR locates a synaptic link, identified as $S_{w1(t),w2(t)}$ if: i) this connection does not exist already; and ii) the ratio between the length of $S_{w1(t),w2(t)}$, computed in input space X as the Euclidean distance between template vectors $T_{w1(t)}$ and $T_{w2(t)}$, and the shortest synapse emanating from either $PE_{w1(t)}$ or $PE_{w2(t)}$ is below user-defined threshold lr. If connection $S_{w1(t),w2(t)}$ is generated, then: iii) synapses emanating from either $PE_{w1(t)}$ or $PE_{w2(t)}$ that do not satisfy CCHR are removed; and iv) the synapse-based (local) selection counter $sbm_{w1(t),w2(t)}$ is initialized to one.

3. If synapse $S_{w1(t),w2(t)}$ exists, its local counter is increased by one, i.e., $sbm_{w1(t),w2(t)}(t) = sbm_{w1(t),w2(t)}(t-1)+1$.

4. Activation values of the PEs belonging to the same map of winner unit $PE_{w1(t)}$, i.e., PEs topologically connected to the winner unit, are computed. Activation values are relative fuzzy membership values computed as

$$R_i(t) = \frac{A_i(t)}{\sum_{h=1}^{c_{w1(t)}} A_h(t)}, \quad i = 1, ..., c_{w1(t)}, \quad R_i(t) \in (0,1), \tag{34}$$

where $A_i(t)$ is an absolute membership value, and $c_{w1(t)}$ is the number of PEs belonging to the same map of winner unit $PE_{w1(t)}$ at presentation time t. Equation (34) shows that $R_i(t)$ is computed on the basis of the absolute memberships of PEs that are topologically connected to $PE_{w1(t)}$; this means that PEs belonging to the same map are coupled through the computation of their relative membership values, while PEs belonging to disjointed maps are computationally independent). Absolute membership $A_i(t)$ is computed as (see Equation (5) in Section 2.1)

$$A_i(t) = \frac{1}{(1 - Gaussian_i(t))^2}, \quad A_i(t) \in (1, \infty). \tag{35}$$

5. FOSART applies a soft competitive learning mechanism, based on $neighborhood-ranking$, to neural units belonging to the same map of best-matching unit $PE_{w1(t)}$. Best ranking $r_{w1(t)} = 0$ is assigned to the best-matching unit $PE_{w1(t)}$. Next, $r_i(t) = 1$ if processing unit PE_i is directly linked to $PE_{w1(t)}$, $r_i(t) = 2$ if processing unit PE_i is indirectly linked to $PE_{w1(t)}$, but directly linked to any processing unit featuring neighborhood-ranking equal to 1, etc.

6. A stochastic (on-line, sequential) weight adaptation law, derived from stochastic optimization of the parameters of a Gaussian mixture model [33], is applied to all processing units belonging to the resonance domain. For every unit PE_i that is topologically connected to the best-matching unit $PE_{w1(t)}$ the adaptation of the receptive field center is computed as follows

$$T_i(t+1) = T_i(t) + \beta_i(t) \cdot (X_k(t) - T_i(t))$$

$$= \frac{1}{\sum_{t^*=1}^{t} \alpha_i(t^*)} \cdot \sum_{t^*=1}^{t} \alpha_i(t^*) X_k(t^*), \quad i = 1, ..., c_{w1(t)}, \quad \forall k \in \{1, n\}. \tag{36}$$

where

$$\beta_i(t) = \frac{\alpha_i(t)}{\sum_{t^*=1}^{t} \alpha_i(t^*)}, \quad i = 1, ..., c_{w1(t)}, \tag{37}$$

$$\alpha_i(t) = \epsilon_i(t) \cdot h_i(t), \quad i = 1, ..., c_{w1(t)}, \tag{38}$$

where learning coefficients $\beta_i(t)$, $\alpha_i(t)$, $\epsilon_i(t)$, and $h_i(t)$ all belong to range (0,1], such that $\alpha_i(t) \leq \min\{\epsilon_i(t),\, h_i(t)\}$, where, in line with Kohonen's first and second learning constraints (see Section 9): i) adaptation step $\epsilon_i(t)$ is expected to decrease monotonically with time; and ii) neighborhood function $h_i(t)$ is expected to: a) monotonically decrease with the lattice distance between units PE_i and $PE_{w1(t)}$; and b) monotonically decrease its spread parameter with time. Inspired by the Neural-Gas algorithm [9], FOSART computes $\epsilon_i(t)$ as

$$\epsilon_i(t) = \epsilon_i(t)(e_i(t), R_i(t)) = \epsilon_{ini}(\epsilon_{fin}/\epsilon_{ini})^{e_i(t)/e_{min}}, \tag{39}$$

where $e_i(t)$ is the PE-based epoch counter at presentation time t, e_{min} is the user-defined decay parameter described in Section 13.1, while variables ϵ_{ini} and ϵ_{fin} are computed as

$$1 \geq \epsilon_{ini} = \max\{R_i(t), \epsilon\} \geq \epsilon_{fin} = \min\{R_i(t), \epsilon\} > 0, \tag{40}$$

where parameter ϵ is the maximum lower limit for the learning rate (e.g., $\epsilon = 0.005$ is fixed by the human designer). Coefficient $\epsilon_i(t)$ is monotonically non-increasing with PE-based (local) epoch counter $e_i(t)$ and monotonically non-decreasing with $R_i(t)$. Owing to the exploitation of $R_i(t)$ in Equation (40), $\epsilon_i(t)$ depends on the set of distances between the input pattern and prototypes belonging to PEs connected to the winner. This dependency between the learning step size and the set of distances between the input pattern and template vectors relates FOSART to the "maximum-entropy" clustering algorithm [9]. In Equation (37), term $h_i(t)$ reduces the overlap between receptive fields according to the following expression, inspired by the Neural-Gas algorithm [9]

$$h_i(t) = h_i(t)[r_i(t), \gamma_i(e_i(t))] = e^{-r_i(t)/\gamma_i(e_i(t))}, \tag{41}$$

where $r_i(t)$ is the neighborhood ranking of node PE_i and $\gamma_i(e_i(t))$ is a spread value computed as a monotonically decreasing function of time, e.g.,

$$\gamma_i(e_i(t)) = \gamma_{ini}(\gamma_{fin}/\gamma_{ini})^{e_i(t)/e_{min}}, \tag{42}$$

where $\gamma_{ini} \geq \gamma_{fin}$. Widely employed settings for these parameters are $\gamma_{ini} = 5$, and $\gamma_{fin} = 0.01$ [9], [64]. Thus, learning coefficient $h_i(t)$ is monotonically decreasing with neighborhood ranking $r_i(t)$ and PE-based epoch counter $e_i(t)$ if $i \neq w1(t)$.

Step 4(b). Non-resonance condition: new processing element allocation. If resonance condition (33) is not satisfied, i.e., "resonance" does not occur and one new processing unit is dynamically allocated to match external expectations. Thus, the PE counter is increased as $c = c + 1$, and a new node PE_c is allocated to match input pattern $X_k t$, such that $T_c(t+1) = X_k(t)$. As a consequence, FOSART requires no randomization of initial templates since initial values are data-driven. Moreover, the PE-based (local) epoch counter $e_c(t)$ is initialized to 0. The PE best-matching counter, bm_c, is initialized to 1.

Step 5. Controls at epoch termination. When the entire input data set is presented to the system, i.e, if $[(t\%n) == 0]$, where operator % computes the remainder of t divided by n, then the following operations occur: a) PE-based time (epoch) counters are incremented by one as $e_i(t+1) = e_i(t) + 1$, $i = 1, ..., c$; b) superfluous cells are removed, such that if PE_i features local counter $bm_i == 0$, $i = 1, ..., c$, i.e., PE_i has not been the best-matching unit for any pattern assignment during the latest processing epoch, then it is removed, and PE counter c is decreased as $c = c - 1$; c) superfluous synaptic links are removed, such that $\forall i \in \{1, c\}$, $\forall h \in \{1, c\}$, if synapse $S_{i,h}$ exists and features $sbm_{i,h} == 0$, i.e., synapse $S_{i,h}$ has not been selected by any pattern assignment during the latest processing epoch, then it is removed; d) all PE-based local counters bm_i, $i = 1, ..., c$, are reset to zero; and e) all synapse-based local counters $sbm_{i,h}$, $\forall i \in \{1, c\}$, $\forall h \in \{1, c\}$, are reset to zero.

Step 6. If PE-based local counter $e_i(t) \geq e_{min}$, $i = 1, ..., c$, then stop. Otherwise, goto Step 1.

13.3 Limitations

• Since FOSART does not minimize any known objective function, its termination is not based on optimizing any model of the process or its data [7].
• FOSART is order-dependent due to on-line learning and example-driven neuron generation.
• It combines mini-batch learning techniques (for neuron generation and removal, and for synapse removal) with example-driven generation of synapses.
• FOSART employs one up to three parameters defined by the human designer rather than by the user. According to [28], these parameters can be considered as significant prior structures (see Section 4.3), i.e., an important property of the model that must be "hard-wired or built-in, perhaps to be tuned later by experience, but not learned in any statistically meaningful way" [71].
• In several tests regarding satellite image clustering that are in progress, FOSART features Mean Square Error values larger than those obtained with SOM [72].

13.4 Advantages

• Owing to its soft competitive implementation, FOSART is expected to be less prone to being trapped in local minima and less likely to generate dead units than hard competitive alternatives [9], [27], [62]. In the tests provided, the system is stable with respect to small changes in input parameters and in the order of the presentation sequence [28], [29].
• Due to its neuron removal strategy, it is robust against noise, i.e., it avoids overfitting.
• Feed-back interaction between attentional and orienting subsystems allows FOSART to self-adjust its size depending on the complexity of the clustering task.

• Distinct sample vectors are employed to initialize reference vectors. This choice reduces the risk of dead unit formation and may reduce computation time with respect to random initialization.
• In the tests provided involving the 2-D Simpson data set (24 patterns) [8], the 2-D two-spirals data set (194 patterns, 2 classes) [73], a 3-D digitized human face (9743 patterns) [18], and the 4-D IRIS data set (150 patterns, 3 classes), FOSART performances are competitive with or better than those of other clustering models found in the literature (VQ, Fuzzy Min-Max, FCM, FLVQ, FALVQ, GLVQ-F, ING). For example, typical error rates for unsupervised categorization of the IRIS data set are 10-16 mistakes [6]. In this case FOSART, scoring 11 mismatches, performs better than: i) the Fuzzy Min-Max neural network model, where the smallest number of misclassified patterns is 18 when the number of clusters is 3 [8]; ii) the off-line Fuzzy c-means algorithm affected by 15 misclassifications [74]; iii) the on-line Kohonen VQ algorithm affected by 17 misclassifications [74]; iv) the class of on-line FALVQ algorithms, affected by 16 misclassifications [54]; and v) the on-line GLVQ-F algorithms, affected by 16 misclassifications [59].
• FOSART is computationally efficient because its computation time increases linearly as ($c+$ No. of links).
• FOSART can be employed in: a) vector quantization, b) density function estimation, and c) structure detection in input data to be mapped in a topologically correct way onto submaps of an output lattice pursuing dimensionality reduction.

13.5 Architectural features

The main features of FOSART are summarized in Table 2.

14 Conclusions

Comparison of clustering models, synthesized in Table 2, shows that SOM, GNG and FOSART develop a soft competitive adaptation strategy, i.e., these models satisfy the definition of fuzzy clustering network adopted herein. Paradoxically, among these architectures only FOSART exploits fuzzy set-theoretic concepts (such as absolute and relative fuzzy membership functions). Fuzzy ART employs a hard competitive adaptation mechanism. FLVQ is affected by asymptotic misbehaviors. Since GNG and FOSART inherit several features from SOM, the importance of SOM is outstanding. Several conclusions stem from our critical review.

1) The "fuzziness" of SOM shows that a network exploiting local rules derived from neurophysiological evidence can verify the definition of fuzzy clustering network proposed in this work, although no fuzzy set-theoretic concept is explicitly adopted.

2) An effective soft competitive strategy makes PEs mutually coupled in terms of both competitive and cooperative contextual effects; in architectural terms, PEs mutually coupled on the basis of their context feature horizontal (intra-layer) connections. While intra-layer connections are found in every biological cognitive system, their explicit adaptation has been introduced in artificial neural networks only recently [49].

3) Within the framework of the multi-disciplinary science of artificial complex systems [44], fuzzy set-theoretic concepts may be adopted to model soft competitive learning mechanisms that are considered useful by the human developer, although they may be difficult to justify on the basis of objective probabilities. These useful mechanisms may be identified in advance by neurophysiological studies of biological neural systems (see point (1) above).

Table 2:

Functional features	**Clustering Algorithms**				
	SOM	FLVQ	Fuzzy ART	GNG	FOSART
network-based (global) variables	time,	time	time, lear.rate	lear.rate	RFS
neuron-based variables	RFC lear.rate	RFC, lear.rate	RFC	RFC, accum.error	RFC, accum.error, age, lear.rate
connection-based variables	N	N	N	age	age
coop./competitive RFC adaptation	SC	SC[1]	HC	SC	SC
on-line/batch RFC adaptation	on-line	batch	on-line	on-line	on-line
neuron generation strategy	N	N	example-driven	mini-batch	example-driven
neuron removal strategy	N	N	N	mini-batch	mini-batch
link generation strategy	N	N	N	example-driven	example-driven
link removal strategy	N	N	N	mini-batch	mini-batch
neuron receptive field information	RFC	RFC	RFC	RFC	RFC, RFS
inter-pattern distance or similarity measure	Eucl.dist.	inverse Eucl.dist.	peculiar[2]	Eucl.dist.	Gaussian basis funct.
size of the neuron receptive field	NB	NB	UB	NB	UB
learning rate	MTD	peculiar[3]	TCO	TCO	MTD
size of the update neighborhood	MTD	TD	N	TCH[4]	TCH[5]
objective function minimization	N	N	N	N	N
fuzzy set-theoretic concepts	N	AMF, RMF	AMF	N	AMF, RMF
econet	N	N	Y	N	Y
prior modular structure	N	N	Y	N	Y
self-organizing modular structure[6]	N[7]	N	N	Y	Y
vector quantization	Y	Y	Y	Y	Y
density estimation	N	N	N	Y	Y
TPM	N[8]	N	N	Y	Y

Legenda
N = No; Y = Yes;
SC = Soft Competitive (cooperative and competitive); HC = Hard Competitive (purely competitive);
NB = Not-Bounded; UB = Upper-Bounded;
RFC = Receptive Field Center; RFS = Receptive Field Spread;
AMF = Absolute Membership Function; RMF = Relative Membership Function;
MTD = Monotonically Time Decreasing; TCO = Constant in Time; TCH = Changes with Time;
TPM = Topology Preserving Mapping;
[1]: FLVQ's learning strategy may be affected by asymptotic misbehaviors.
[2]: Fuzzy ART employs an interpattern similarity measurement that is not symmetrical.
[3]: In descending FLVQ, for increasing training epochs the learning rate of the winner unit tends to increase (see Section 10.2).
[4]: Resonance neighborhood consists of neurons directly connected to the winner.
[5]: Resonance neighborhood consists of neurons belonging to the same map as the winner.
[6]: generation of disjointed maps.
[7]: SOM maps different input patterns into spatially segregated areas of the output lattice, i.e., these segregated areas are not topologically disjointed.
[8]: SOM performs TPM iff the dimension of the input space is not larger than the dimension of the output lattice.

References

[1] A. Baraldi and F. Parmiggiani, "Fuzzy clustering: critical analysis of the contextual mechanisms employed by three neural network models," in *SPIE Proc. on Applications of Fuzzy Logic Technology III*, B. Bosacchi, J. Bezdek, Eds., SPIE vol. 2761, pp. 261-270, 1996.

[2] Anonymous referee, *IEEE Trans. on Neural Networks*, 1997.

[3] S. Mitra and S. K. Pal, "Self-organizing neural network as a fuzzy classifier," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 385-399, 1994.

[4] T. Kohonen,"The self-organizing map," *Proceedings of the IEEE* vol. 78, no. 9, pp. 1464-1480, 1990.

[5] T. Kohonen, *Self-organizing Maps.* 2nd Edition, Springer Verlag, Berlin, 1997.

[6] J. C. Bezdek and N. R. Pal, "Two soft relative of learning vector quantization," *Neural Networks*, vol. 8, no. 5, pp. 729-743, 1995.

[7] E. C. Tsao, J. C. Bezdek and N. R. Pal.,"Fuzzy Kohonen clustering network," *Pattern Recognition*, vol. 27, no. 5, pp. 757-764, 1994.

[8] P. K. Simpson,"Fuzzy Min-Max neural network-Part 2: Clustering," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 32-45, 1993.

[9] T. M. Martinetz, S. G. Berkovich and K. J. Schulten," Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 558-569, 1993.

[10] H. Ritter, T. M. Martinetz, and K. J. Schulten, *Neural computation and self-organizing maps.* Reading, MA: Addison-Wesley, 1992.

[11] M. Porat and Y. Zeevi, "The generalized Gabor scheme of image representation in biological and machine vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 4, pp. 452-467, July 1988.

[12] K. Rose, F. Guerewitz, and G. Fox, "Statistical mechanics and phase transitions in clustering," *Physical Rev. Letters*, vol. 65, no. 8, pp. 945-948, 1990.

[13] K. Rose, F. Guerewitz, and G. Fox, "A deterministic approach to clustering," *Pattern Recognition Letters*, vol. 11, no. 11, pp. 589-594, 1990.

[14] S. P. Luttrell, "Derivation of a class of training algorithms," *IEEE Trans. on Neural Networks*, vol. 1, pp. 229-232, 1990.

[15] S. P. Luttrell, "A Bayesian analysis of self-organizing maps", *Neural Computation*, vol. 6, pp. 767-794, 1994.

[16] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[17] H. Kumazava, M. Kasahara, and T. Namekawa, "A construction of vector quantisers for noisy channels," *Elect. Eng. Jpn.*, vol. 67B, pp. 39-47, 1984.

[18] M. Fontana, N. A., Borghese, and S. Ferrari, "Image reconstruction using improved Neural-Gas," in *Proc. Italian Workshop on Neural Networks '95*, M. Marinaro and R. Tagliaferri Eds. Singapore: World Scientific, pp. 260-265, 1995.

[19] G. Carpenter, S. Grossberg and D.B. Rosen, "Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991.

[20] G. Carpenter, S. Grossberg, N. Maukuzon, J. Reynolds and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 698-713, 1992.

[21] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

[22] G. A. Carpenter and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, no. 23, pp. 4919-4930, 1987.

[23] C. Hung and S. Lin, "Adaptive Hamming Net: a fast-learning ART 1 model without searching," *Neural Networks*, vol. 8, no. 4, pp. 605-618, 1995.

[24] A. Baraldi and E. Alpaydın, "Simplified ART: A new class of ART algorithms," International Computer Science Institute, Berkeley, CA, TR-98-004.

[25] A. Baraldi and F. Parmiggiani, "Fuzzy combination of the Kohonen and ART neural network models to detect statistical regularities in a random sequence of multi-valued input patterns," in *Proc. Int. Conf. on Neural Networks '97*, Houston, TX, vol. 1, pp. 281-286, 1997.

[26] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, pp. 625-632, 1995.

[27] B. Fritzke, "Some competitive learning methods", Draft document, *http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG*, 1998.

[28] A. Baraldi and F. Parmiggiani, "A fuzzy neural network model capable of generating/removing neurons and synaptic links dynamically," in *Proc. of the II Italian Workshop on Fuzzy Logic*, Bari, Italy, March 1997, P. Blonda, M. Castellano and A. Petrosino, Eds. Singapore: World Scientific, 1998.

[29] A. Baraldi and F. Parmiggiani, "Novel neural network model combining radial basis function, competitive Hebbian learning rule, and fuzzy simplified adaptive resonance

theory", in *Proc. SPIE's Optical Science, Engineering and Instrumentation '97: Applications of Fuzzy Logic Technology IV*, San Diego, CA, vol. 3165, pp. 98-112, 1997.

[30] Y. Pao, *Adaptive pattern recognition and neural networks.* Reading, MA: Addison-Wesley, 1989.

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statist. Soc. Ser. B*, vol. 39, pp. 1-38, 1977.

[32] E. Alpaydın, "Soft vector quantization and the EM algorithm," *Neural Networks*, 1998, in press.

[33] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford, UK: Clarendon Press, 1995.

[34] C. M. Bishop, M. Svensen, and C. Williams, "GTM: a principled alternative to the self-organizing map," in *Proc. Int. Conf. on Artificial Neural Networks, ICANN'96.* Springer-Verlag, pp. 164-170, 1996.

[35] V. Cherkassky and F. Mulier, *Learning From Data: Concepts, Theory, and Methods.* New York: Wiley, 1998.

[36] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer-Verlag, 1995.

[37] C. Burges, "A tutorial on Support Vector Machines for pattern recognition," submitted to *Data Mining and Knowledge Discovery*, 1998.

[38] M. J. Jordan and R. A. Jacobs, "Hierarchical mixture of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181-214, 1994.

[39] M. J. Jordan and C. M. Bishop, "An introduction to graphical models and machine learning," draft document, 1998.

[40] M. Bianchini and M. Gori, "Optimal learning in artificial neural networks: a review of theoretical results," *Neurocomputing*, vol. 13, no. 2-4, pp. 313-346, 1996.

[41] M. Bianchini, P. Frasconi, M. Gori, "Learning without local minima in radial basis function networks," *IEEE Trans. Neural Networks*, vo. 6, no. 3, pp. 749-756, 1995.

[42] T. Mitchell, *Machine Learning.* New York: McGraw-Hill, 1997.

[43] J. Buhmann, "Learning and data clustering," in *Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Bradford Books / MIT Press, 1995.

[44] R. Serra and G. Zanarini, *Complex systems and cognitive processes.* Berlin, Germany: Springer-Verlag 1990.

[45] R. Llinas and P. Churchland, *The Mind-Brained Continuum.* Cambridge, MA: MIT Press, 1996.

[46] B. Happel and J. Murre, "Design and evolution of modular neural network architecture," *Neural Networks*, vol. 7, no. 6/7, pp. 985-1004, 1994.

[47] M. L. Minsky and S. A. Papert, *Perceptrons - Expanded Edition.* Cambridge, MA: MIT Press, 1988.

[48] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge MA: MIT Press, 1986.

[49] T. M. Martinetz and K. J. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, no. 3, pp. 507-522, 1994.

[50] D. Parisi, "La scienza cognitiva tra intelligenza artificiale e vita artificiale," in *Neuroscienze e Scienze dell'Artificiale: dal Neurone all'Intelligenza* E. Biondi et al., Eds. Bologna, Italy: Patron, pp. 321-341, 1991.

[51] Y. Zheng and J. F. Greenleaf, "The effect of concave and convex weight adjustments on self-organizing maps," *IEEE Trans. on Neural Networks*, vol. 7, no. 1, pp. 87-96, 1996.

[52] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. National Academy of Sciences*, vol. 74, pp. 2554-2558, 1982.

[53] T. Masters, *Signal and Image Processing With Neural Networks. A c++ Sourcebook.* New York: Wiley, 1994.

[54] N. B. Karayiannis and P. Pai, "Fuzzy algorithms for learning vector quantization," *IEEE Trans. on Neural Networks*, vol. 7, no. 5, pp. 1196-1211, 1996.

[55] N. B. Karayiannis, and M. Ravuri, "An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering", in *Intelligent Engineering Systems Through Artificial Neural Networks*, C. H. Dagli et al., Eds., vol. 5, New Yoork, NY: ASME Press, pp.247-252,1995.

[56] N. B. Karayiannis, J. C. Bezdek, "An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering", *IEEE Trans. on Fuzzy Systems*, vol. 5, no. 4, pp. 622-628, 1997.

[57] N. B. Karayiannis, " Learning vector quantization: A review," *Int. Journal of Smart Engineering System Design*, vol. 1, pp. 33-58, 1997.

[58] J. C. Bezdek, and N. R. Pal," Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 549-557, 1993.

[59] N. B. Karayiannis, J. C. Bezdek, N. R. Pal, R. J. Hathaway and P. Pai, "Repair to GLVQ: A new family of competitive learning schemes, *IEEE Trans. on Neural Networks*, vol. 7, no. 5, pp. 1062-1071, 1996.

[60] A. Baraldi, P. Blonda, F. Parmiggiani, G. Pasquariello and G. Satalino, "Model transitions in descending FLVQ," *IEEE Trans. on Neural Networks*, vol. 9, no. 5, 1998.

[61] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 2, pp. 98-110, 1993.

[62] R. N. Davè and R. Krishnapuram, "Robust clustering method: a unified view," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 2, pp. 270-293, 1997.

[63] M. Barni, V. Cappellini, and A. Mecocci, "Comments on a possibilistic approach to clustering," *IEEE Trans. Fuzzy Systems*, vol. 4, no. 3, pp. 393-396, 1996.

[64] F. Ancona, S. Ridella, S., Rovetta, and R. Zunino, "On the importance of sorting in Neural Gas training of vector quantizers," in *Proc. International Conference on Neural Networks '97*, Houston, TX, June 1997, vol. 3, pp. 1804-1808, 1997.

[65] E. Erwin, K., Obermayer, and K. Schulten, "Self-organizing maps: ordering, convergence properties and energy functions," *Biol. Cybernetics*, vol. 67, pp. 47-55, 1992.

[66] J. R. Williamson, "Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, no. 5, pp. 881-897, 1996.

[67] J. R. Williamson, "A constructive, incremental-learning network for mixture model ing and classification," *Neural Computation*, vol. 9, pp. 1517-1543, 1997.

[68] ftp://ftp.sas.com/pub/neural/FAQ.

[69] F. Y. Shih, J. Moh and F. Chang, " A new ART-based neural architecture for pattern classification and image enhancement without prior knowledge," *Pattern Recognition*, vol. 25, no. 5, pp. 533-542, 1992.

[70] J. Huang, M. Georgiopoulos and G. L. Heileman, "Fuzzy ART properties," *Neural Networks*, vol. 8, no. 2, pp. 203-213, 1995.

[71] S. Geman, E. Bienenstock and R. Dourstat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1-58, 1992.

[72] P. Blonda, G. Satalino, A. Baraldi and A. Bognani, "Neuro-fuzzy analysis of remote sensed antarctic data," in Proc. of the II Italian Workshop on Fuzzy Logic, Bari, Italy, March 1997, P. Blonda, M. Castellano, A. Petrosino, Eds. Singapore: World Scientific, 1998.

[73] B. Fritzke, "Growing cell structures - A self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441-1460, 1994.

[74] S. Kim and and S. Mitra, "Integrated Adaptive Fuzzy Clustering (IAFC) algorithm," in *Proc. of the Second IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1264-1268, 1993.

[75] B. Fritzke. Personal communication, 1997.

[76] B. Fritzke., "The LBG-U method for vector quantization - An improvement over LBG inspired from neural networks," *Neural Processing Letters*, vol. 5, no. 1, 1997.