INTERNATIONAL COMPUTER SCIENCE INSTITUTE

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

Soft-to-hard model transition in clustering: a review

A. Baraldi* and L. Schenato†

TR-99-010

September 1999

Abstract

Clustering analysis often employs unsupervised learning techniques originally developed for vector quantization. In this framework, a frequent goal of clustering systems is to minimize the *quantization error*, which is affected by many local minima. To avoid confinement of reference vectors to local minima of the quantization error and to avoid formation of dead units, hard c-means clustering algorithms are traditionally adapted by replacing their hard competitive strategy with a soft adaptation rule, where the degree of overlap between receptive fields is proportional to a monotonically decreasing scale (temperature) parameter. By starting at a high temperature, which is carefully lowered to zero, a soft-to-hard competitive clustering model transition is pursued, such that local minima of the quantization error are expected to emerge slowly, thereby preventing the set of reference vectors from being trapped in suboptimal states. A review of the hard c-means, Maximum-Entropy, Fuzzy Learning Vector Quantization (FLVQ), Neural Gas (NG), Self-Organizing Map (SOM) and a mixture of Gaussians method is provided, relationships between these methods are highlighted and a possible criterion for discriminating between different soft-to-hard competitive clustering model transitions is suggested.

*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

†Dipartimento di elettronica e informatica Universita' degli Studi di Padova, DEI - Via Gradenigo 6/a, 35131 Padova, Italy, Ph.: +39+049+827-7500, Fx.: +39+049+827-7699, lusche@robotics.eecs.berkeley.edu

1 Introduction

Given a presentation sequence of unlabeled multidimensional patterns $X_i \in \mathcal{R}^n$, $i = 1, ..., m$, unsupervised learning systems detect a set of parameters capable of modeling hidden data structures (e.g., linear substructures), statistical data regularities, or probability density functions [1]. Among unsupervised learning tasks, the problem of clustering is that of separating the unlabeled data set into groups (i.e., hidden data structures), called clusters, for which samples within a cluster are more similar than samples from different clusters. Usually, a vector prototype, also called reference or template vector, $C_j \in \mathcal{R}^n$, $j = 1, ..., c \leq m$, is generated to characterize the members of a cluster as a group. Since the goal of clustering is to group the data at hand rather than provide an accurate characterization of unobserved (future) samples generated from the same probability distribution, the task of clustering can fall outside the framework of predictive (inductive) learning. In spite of this, clustering analysis often employs unsupervised learning techniques originally developed for vector quantization, which is a predictive learning problem [2].

In this framework, a frequent goal of clustering systems is the minimization of the *distorsion error*, also called *reconstruction error* [3], or *quantization error* [1], defined as

$$E_{rec} = \frac{1}{m} \sum_{i=1}^{m} ||X_i - C_{w1(i)}||^2, \quad w1(i) \in \{1, c\}, \tag{1}$$

where $||\cdot||$ identifies the Euclidean distance, c is the number of processing units in the system (network) and $w1(i)$ is the index of the best-matching template $C_{w1(i)}$ detected as

$$||X_i - C_{w1(i)}|| \leq ||X_i - C_j||, \quad w1(i) \in \{1, c\}, j = 1, ..., c.$$

Eq. (1) can be considered the hard competitive (Winner-Takes-All, WTA) case of the more general *distance-weighted sum-of-squares clustering cost function* (see Eq. (2) below) [4]. In Eq. (1), reference vectors are the free parameters (degrees of freedom) of the unsupervised learning task which has many local minima in the parameter space. Minimization of Eq. (1) is called a hard (competitive) c-means clustering problem [4], [5]. In this case, Eq. (1) provides each reference vector with a region of support which is a subset of the data space known as a Voronoi polyhedron, the whole set of reference vectors providing a partition of the input space known as Voronoi tesselation [1], [6]. Voronoi tessellation is one-to-one related to Delaunay triangulation, which is a peculiar form of triangulation in various geometrical and functional respects, e.g., Delaunay triangulation has been proved to be optimal for piecewise linear function approximation over a triangulation of the input samples [7], [8]. In terms of data structure detection, Eq. (1) performs well for certain kind of data, but its failure at detection of non-convex data structures, e.g., linear substructures or bananas, is well documented [9].

Unsupervised learning systems whose goal is to minimize Eq. (1) are called vector quantizers and the set of reference vectors C_j, $j = 1, ..., c$ is called a codebook [1]. In a vector quantization task, data is transmitted over a (limited bandwith) communication channel by transmitting, for each input data vector, only the index of one reference vector. The set of reference vectors (codebook) is assumed to be known both to sender and receiver. Therefore, the receiver can use the sequence of received indexes to reconstruct the input

data set by retrieving the corresponding reference vectors. If the input data distribution is clustered, i.e., it forms disjointed subregions where probability density values are large, vector quantization provides compact data coding with relatively little distorsion error [1]. According to Eq. (1), the index being sent through the communication channel is that corresponding to the reference vector nearest to the input data vector. This may no longer be the case if the communication channel is assumed to be affected by noise (i.e., if the received index is equal to the index being sent plus noise) [10].

A more general form of Eq. (1), termed *distance-weighted sum-of-squares quantization error*, which can be minimized by an unsupervised learning algorithm is [4] (p. 188), [11], [12], [3],

$$E_{gen} = \sum_{i=1}^{m} \sum_{h=1}^{c} ||X_i - C_h||^2 k_h(d(X_i, \mathbf{C})), \tag{2}$$

where symbol $d(\cdot)$ identifies an interpattern distance metric, $d(X_i, \mathbf{C}) = \{d(X_i, C_1), ..., d(X_i, C_c)\}$ identifies a set of interpattern distances between data point X_i and the whole set of vector prototypes $\mathbf{C} = \{C_1, ..., C_c\}$, while term $k_h(d(X_i, \mathbf{C})) \geq 0$ is a *distance-weighting function*, also termed *kernel function* [2], [13], monotonically non-increasing with distance $d(X_i, C_h)$ and monotonically non-decreasing with distances $d(X_i, C_j)$, $j = 1, ..., c, h \neq j$. Weighting function $k_j(d(X_i, \mathbf{C}))$ models the c processing units, whose receptive field centers are the vector prototypes C_j, $j = 1, ..., c$, as coupled network nodes provided with feed-sideways (lateral) connections which guarantee network-wide internode communication [14]. This node-coupling model reflects the intuitive belief that more information about an input pattern is simultaneously gathered by the whole set of prototypes than by any single prototype in the set [11]. It is typically pursued by implementing the kernel function as a relative (probabilistic) fuzzy membership function [15], [16]. These systems are affected by the so-called relative membership problem, i.e., they are not robust to the presence of noise and outliers (i.e., a noise point can drastically influence the estimate of the prototype). To reduce this drawback, a special case of Eq. (2) is that in which the membership of a point in a cluster is determined solely by how far that point is from the prototype of the class with no regard to other classes, i.e., Eq. (2) becomes

$$k_h(d(X_i, \mathbf{C})) = k(d(X_i, C_h)),$$

i.e., the kernel function computes memberships that are possibilistic or absolute (i.e., not relative) [15], [16]. This leads to systems, e.g., the Possibilistic c-means clustering algorithm [15], where cluster centers are sought independently of one another. These systems are affected by the tendency to produce coincident clusters [16], [17].

Notice that Eq. (2) can also be interpreted as a link between clustering problems and distance-weighted regression [13] (see Appendix 1).

2 Optimization issues

The necessary condition that guarantees approximate minimization of Eq. (2) is

$$\frac{\partial E_{gen}}{\partial C_j} = -\sum_{i=1}^{m} (X_i - C_j) k_j(d(X_i, \mathbf{C})) + R_j = 0, \quad j = 1, ..., c, \tag{3}$$

where

$$R_j = \sum_{i=1}^{m} \sum_{h=1}^{c} (X_i - C_h)^2 \frac{\partial k_h(d(X_i, \mathbf{C})}{\partial C_j}, \quad j = 1, ..., c. \tag{4}$$

If we assume [3] that

$$R_j = 0, \quad j = 1, ..., c, \tag{5}$$

then the iterative batch solution of Eq. (3) becomes

$$C_j^{(e+1)} = \frac{\sum_{i=1}^{m} X_i k_j(d(X_i, \mathbf{C}^{(e)}))}{\sum_{g=1}^{m} k_j(d(X_g, \mathbf{C}^{(e)}))}, \quad j = 1, ..., c, \tag{6}$$

where variable e identifies the number of processing epochs, i.e., the number of times the finite training data set is repeatedly presented to the network. This batch learning process is also called instance-based [18], or memory-based learning [13], because it requires the storage of a complete data set since each data point cannot be discarded once it has been used. Notice that Eq. (6) computes any template vector as a convex combination of the input patterns: since the convex combination (e.g. , the average value) of a non-convex data set may lie well outside the data manifold, it is obvious that Eq. (2) cannot perform well for non-convex types of data.

If the assumption about vanishing term R_j holds, the recursive batch gradient descent solution of Eq. (2) is defined as

$$C_j^{(e+1)} \doteq C_j^{(e)} - \epsilon(e) \frac{\partial E_{gen}}{\partial C_j^{(e)}} = C_j^{(e)} + \epsilon(e) \sum_{i=1}^{m} (X_i - C_j^{(e)}) k_j(d(X_i, \mathbf{C}^{(e)})), \quad j = 1, ..., c. \tag{7}$$

In general, the learning rate $\epsilon(e)$ of the exact gradient descent solution ought to satisfy the three conditions applied to the coefficients of the Robbins-Monro algorithm to find the roots of a function iteratively (in our case, the function whose roots are investigated is $\partial E_{gen}/\partial C_j$). These conditions are [4] (pp. 47, 96), [19],

i) $\lim_{t \to \infty} \epsilon(t) = 0$;
ii) $\sum_{t=1}^{\infty} \epsilon(t) = \infty$;
and iii) $\sum_{t=1}^{\infty} \epsilon^2(t) < \infty$.

For example, when $\epsilon(e) = 1/e$ [4] (p. 96), [12], [1], then $\epsilon(e)$ decreases monotonically with e under Robbins-Monro conditions. Condition iii) states that learning rate $\epsilon(t)$ must decrease fast enough, while condition ii) limits the rate of decrease of the learning rate: indeed, if this rate of decrease is too quick, then it could stop the progression of the algorithm towards the minimum [20]. According to condition ii), the infinite sum of the learning rates diverges. This is tantamount to saying that even after a large number of input signals and correspondingly low values of the learning rate $\epsilon(t)$, arbitrarily large modifications of reference vectors may occur in principal, although they are most unlikely to occur [1]. When the data distribution is stationary, conditions i) to iii) are necessary but not sufficient to guarantee that true (batch) and stochastic (on-line) gradient descent allgorithms converge to a point in the parameter space [20].

A sequential (stochastic, on-line) update process, whose goal is to avoid the storage of all data points by assuming that they are arriving one at a time, can be derived from Eq. (7), when Eq. (5) holds, by dropping the sum over input patterns [4] (p. 96), [21], i.e.,

$$C_j^{(t+1)} = C_j^{(t)} + \epsilon(t)(X^{(t)} - C_j^{(t)}) k_j(d(X^{(t)}, \mathbf{C}^{(t)})), \quad j = 1, ..., c, \tag{8}$$

where $X^{(t)}$ is the current data vector at presentation time t, and learning rate $\epsilon(t)$ is subject to the Robbins-Monro conditions listed above. Implementations of Eq. (8) exploiting different weight expressions $k_j(d(X^{(t)}, \mathbf{C}^{(t)}))$ can be found, for example, in the Learning Vector Quantization (LVQ) algorithm [22], [23], in the Neural Gas model [3], and in the Maximum-Entropy clustering method [3].

To derive an explicit formula of learning rate coefficient $\epsilon(t)$ in the sequential update mode, an approach alternative to Eq. (8) is to separate out from Eq. (6) the contribution from the $(m+1)$th data point [4] (p. 46), [24]. This gives

$$C_j^{(t+1)} = C_j^{(t)} + \beta_j^{(t)} \cdot (X^{(t)} - C_j^{(t)}), \quad j = 1, ..., c, \tag{9}$$

where

$$\beta_j^{(t)} = \frac{k_j(d(X^{(t)}, \mathbf{C}^{(t)}))}{\sum_{T=1}^{t} k_j(d(X^{(T)}, \mathbf{C}^{(T)}))}, \tag{10}$$

i.e., Eqs. (9) and (10) are a special case of Eq. (8) when

$$\epsilon(t) = \frac{1}{\sum_{T=1}^{t} k_j(d(X^{(T)}, \mathbf{C}^{(T)}))}. \tag{11}$$

Eqs. (9) and (10) can also be written as [1]

$$\begin{aligned} C_j^{(t+1)} &= C_j^{(t)} + \frac{k_j(d(X^{(t)}, \mathbf{C}^{(t)}))}{\sum_{T=1}^{t} k_j(d(X^{(T)}, \mathbf{C}^{(T)}))} \cdot (X^{(t)} - C_j^{(t)}) \\ &= \frac{\sum_{T=1}^{t} X^{(T)} k_j(d(X^{(T)}, \mathbf{C}^{(T)}))}{\sum_{T=1}^{t} k_j(d(X^{(T)}, \mathbf{C}^{(T)}))}, \quad j = 1, ..., c. \end{aligned} \tag{12}$$

Notice the similarity between Eqs. (6) and (12) [1]. Interpattern distance $d(\cdot)$ can be computed according to the L_p norm (Minkowsi metric) which, in a general multidimensional case, is

$$d(A, B) = \left(\sum_{q=1}^{n} |a_q - b_q|^p \right)^{\frac{1}{p}}, \quad \forall A, \forall B \in \mathcal{R}^n. \tag{13}$$

If $p = 2$, Eq. (13) computes the Euclidean distance. Kernel function $k_j(d(X, \mathbf{C})), \forall X \in \mathcal{R}^n$, $\forall C_j \in \mathcal{R}^n$, $j = 1, ..., c,$, usually (but not always) satisfies the following properties (adapted from [2])

$k_j(d(X, \mathbf{C})) > 0.$	Positive.	$(14a)$
$k_j(d(X, \mathbf{C})) = k_j(d(\mathbf{C}, X)).$	Radially symmetric.	$(14b)$
$k_j(d(X, \mathbf{C})) =$ max iff $X = C_j.$	Takes on its maximum when $X = C_j.$	$(14c)$
$\lim_{d \to \infty} k_j(d) = 0.$	Localized function [4].	$(14d)$

If the kernel function satisfies conditions (14a) to (14d), then Eq. (2) becomes the *soft competitive version of the distance-weighted sum-of-squares clustering cost function.* Indeed, according to Eqs. (6) and (12), any input pattern affects all vector protoypes to different degrees depending on their proximity to the input pattern. In other words, there is overlapping between regions of support of the processing elements, such that, given the activation function $f_j(X)$ of the jth processing unit, $j = 1, ..., c$, and given a threshold ϵ arbitrarily close to zero, the region of support of function $f_j(X)$ is the input subset $\{X\}_j$ where $f_j(X) > \epsilon$ holds true. Local regions that overlap are commonly called fuzzy [2].

If the kernel function is such that

$$k_{w1(X)}(d(X, \mathbf{C})) = \begin{cases} 1, \text{ if } d(X, C_{w1(X)}) \leq d(X, C_h), \forall X \in \mathcal{R}^n, w1(X) \in \{1, c\}, h = 1, ..., c, \\ 0, \text{ otherwise,} \end{cases} \tag{15}$$

where node index $w1(X)$ identifies the best-matching unit for pattern X, then Eq. (15) does not satisy constraint (14a). When it employs weighting values that satisfy Eq. (15), Eq. (2) becomes equivalent to Eq. (1). When weighting values are computed with Eq. (15) where $d(\cdot) = ||\cdot||$ (Euclidean distance), then Eqs. (6) and (12) become, respectively, the batch (Lloyd's or Forgy's [5]) and on-line (McQueen's [4], [5], [25]) iterative solution of the well-know hard c-means clustering problem.

It can be proved that when Eq. (15) employs $d(\cdot) = ||\cdot||$ (Euclidean distance), then terms R_j, $j = 1, ..., c$, vanish in Eq. (4), i.e., Eqs. (5) to (10) hold true (see Appendix 2). Moreover, notice that when Eq. (15) is substituted into Eq. (1), then for any template vector $C_j^{(t)}$ which is not a dead unit the following relation holds

$$\sum_{t=1}^{\infty} \beta_j(t) = 1 + 1/2 + 1/3 + ... = \sum_{m=1}^{\infty} 1/m,$$

i.e., due to the properties of the harmonic series, which is not summable but square summable, these learning rate values satisfy the Robbins-Monro criteria.

If, besides constraints (14a) to (14d) the kernel function satisfies conditions

$$0 < k_j(d(X, \mathbf{C})) \leq 1, \quad j = 1, ..., c, \forall X \in \mathcal{R}^n, \tag{16}$$

$$\sum_{j=1}^{c} k_j(d(X, \mathbf{C})) = 1, \quad \forall X \in \mathcal{R}^n, \tag{17}$$

then the kernel function is a relative or probabilistic fuzzy membership function [26], [27], [15], [16]. Notice that Eq. (17) provides a tool for modeling network-wide internode communication by assuming that the c processing elements, whose receptive field centers are the vector prototypes C_j, $j = 1, ..., c$, are coupled through feed-sideways (lateral) connections [14]. From a functional standpoint, distributed systems like, for example, Fuzzy Learning Vector Quantization (FLVQ) [11], [12], and the Estimation-Maximization (EM) algorithm [28] applied to Gaussian mixtures [4] where, respectively, a "useful" relative fuzzy membership function or an objective posterior probability estimate is computed, are equivalent to distributed systems where a contextual (competitive and cooperative) effect mechanism

employing feed-sideways (intra-layer) connections is modeled explicitly. Notice that only few clustering networks employ intra-layer connections explicitly, i.e., by means of specific data structures and parameter adaptation strategies [1], [26], [29], [30]. As already mentioned, the relative membership problem affects all these systems unless special mechanisms for noise and outlier detection are adopted as in [27], [30].

About the choice of the kernel function constrained by Eqs. (14a) to (14d) and Eqs. (16) and (17), considerations similar to those applied to distance-weighted regression methods can be adopted [2], [13] (see Appendix 1).

When the kernel function is constrained by Eqs. (14a) to (14d) and Eq. (17), but condition $0 < k_j(d(X, \mathbf{C})) < 1$ holds in place of Eq. (16) when $d(X, C_j) = 0$, i.e., when $X = C_j$, then, according to Eqs. (6) and (12), all templates are attracted by an input pattern X despite the fact that this pattern has been perfectly matched by one reference vector. This behavior is accomplished by choosing, for example,

$$k_{i,j;1} = k_j^2(\sigma, d(X_i, \mathbf{C})) = \frac{e^{-\frac{d(X_i, C_j)^2}{\sigma^2}}}{\sum_{h=1}^{c} e^{-\frac{d(X_i, C_h)^2}{\sigma^2}}}, \quad j = 1, ..., c, \quad i = 1, ..., m, \tag{18}$$

where $\sigma > 0$ is a scale parameter. When $d(\cdot) = ||\cdot||$, Eq. (18) becomes the weighting function adopted by the Maximum-Entropy clustering algorithm, which is traditionally employed in data transmission and coding [3]. The cost function of the Maximum-Entropy clustering algorithm is the negative log-likelihood for the data set modeled as a Gaussian mixture where spread parameter *sigma* is fixed (see also Eq. (34)), i.e., [3]

$$E_{mec}(\mathbf{C}, \sigma) = -\sigma^2/2 \sum_{i=1}^{m} ln \left[\sum_{j=1}^{c} e^{-\frac{||X_i - C_j||^2}{\sigma^2}} \right]. \tag{19}$$

It is easy to prove that, to minimize Eq. (19), necessary condition $\partial E_{mec}/\partial C_j = 0$, $j = 1, ..., c$, provides a batch update equation equivalent to

$$C_j^{(e+1)} = \frac{\sum_{i=1}^{m} X_i k_{i,j;1}}{\sum_{g=1}^{m} k_{i,g;1}}, \quad j = 1, ..., c, \tag{20}$$

where Eq. (20) is a special case of Eq. (6). The on-line version of Eq. (20) is a special case of either Eq. (8) or Eq. (12) where membership values (18) are employed.

Notice that if $\sigma \to 0$, then Eq. (18) becomes equivalent to the WTA kernel function (15) and Eq. (19) reduces to Eq. (1). In this limit case, if $d(\cdot) = ||\cdot||$ (Euclidean distance), then receptive fields of processing elements are Voronoi polyhedra that feature no degree of overlap. See Appendix 2 to examine how Eq. (18) relates to Eq. (5). Since Eq. (18) does not satisfy Eq. (5) whatever scale parameter σ may be, then cost function (19) cannot be considered as a special case of Eq. (2) for which Eqs. (3) to (12) hold true.

Rose *et al.* [31] have shown that the form of Eq. (19) suggests that the parameter update strategy given by Eqs. (18) and (20) is, in fact, a deterministic annealing procedure with σ as the temperature and E_{mec} as the free energy [3]. The temperature controls the amplitude of noise artificially introduced in the optimization process [24], such that at a high temperature regime Eq. (19) is convex (i.e., without local minima). By starting at a high temperature which, during the adaptation process, is carefully lowered to zero, cost

function (1) is recovered while its local minima are avoided [3]. In theory, $\sigma \propto 1/\ln t$, where t is the number of sequential pattern presentations, must hold.

Finally, notice that the Maximum-Entropy clustering algorithm is affected by the relative membership problem.

Analogously to the Maximum-Entropy clustering algorithm, other unsupervised learning systems found in the literature employ soft-to-hard learning strategy transitions aiming at progressively reducing the degree of overlap between receptive fields until a Voronoi tessellation of the input space is reached. By gradually decreasing a spread parameter the cost function minimized by these systems must become equivalent to Eq. (1), i.e., the local minima of Eq. (1) are expected to emerge gradually, therefore preventing the set of reference vectors from being trapped in suboptimal states. Examples of these systems are the on-line Self-Organizing Map (SOM) [22], [23], on-line Neural Gas algorithm [3], and the batch Fuzzy Learning Vector Quantization (FLVQ) model [11], [12]. Notice that if the ultimate goal of soft to hard learning strategy transitions is to minimize Eq. (1), this is tantamount to saying that these clustering algorithms are designed to work basically as vector quantizers.

Differently from the behavior of kernel function (19), notice that when Eqs. (16) and (17) are adopted, if $k_j(d(X,\mathbf{C})) = 1$ when $d(X,C_j) = 0$, i.e., when $X = C_j$, then $k_h(d(X_i,\mathbf{C})) = 0$, $h = 1,...,c$, $h \neq j$, i.e, when a pattern is perfectly matched by one template vector, then this pattern does not affect any other template in the network according to Eqs. (6) and (12). This behavior is accomplished by choosing, for example,

$$k_{i,j;2} = k_j(p, d(X_i,\mathbf{C})) = \frac{\frac{1}{d(X_i,C_j)^{2/p}}}{\sum_{h=1}^{c} \frac{1}{d(X_i,C_h)^{2/p}}}, \quad j = 1,...,c, \quad i = 1,...,m, \tag{21}$$

where $p > 0$ is a scale parameter. This relative membership function was presented in the context of fuzzy clustering algorithms such as Fuzzy $c-$means (FCM), where $d(\cdot) = ||\cdot||$ and scale parameter p is fixed, and Fuzzy Learning Vector Quantization (FLVQ), where scale parameter p decreases monotonically with the number of training epochs [11], [12]. The FCM/FLVQ cost function is defined as

$$E_{FLVQ}(\mathbf{K},\mathbf{C},p) = \sum_{i=1}^{m}\sum_{j=1}^{c} ||X_i - C_j||^2 (k_{i,j;2})^{p+1}, \tag{22}$$

where $\mathbf{K}$ is a fuzzy c-partition of the input data set such that coefficient $k_{i,h;2}$ is a relative number equal to the degree of similarity between pattern X_i and template C_j subject to constraint (17). Assuming $p = cost$, the Lagrangian function corresponding to Eq. (22) is [32]

$$E_{FCM}(\mathbf{K},\mathbf{C},\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{c} ||X_i - C_j||^2 (k_{i,j;2})^{p+1} + \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{c} k_{i,j;2} - 1 \right). \tag{23}$$

By differentiating Eq. (23) with respect to C_j (for fixed membership values and parameter p), we get

$$C_j^{(e+1)} = \frac{\sum_{i=1}^{m} X_i k_{i,j;2}}{\sum_{g=1}^{m} k_{i,g;2}}, \quad j = 1,...,c, \tag{24}$$

where Eq. (24) is a special case of Eq. (6). By letting $w_{i,j}^2 = k_{i,j;2}$ and differentiating Eq. (23) with respect to $w_{i,j}$ (for fixed cluster templates and parameter p) we get

$$\frac{\partial E_{FCM}(\mathbf{W},\mathbf{C},\lambda)}{\partial w_{i,j}} = 2(p+1)w_{i,j}^{2(p+1)-1}(X_i - C_j)^2 + 2w_{i,j}\lambda_i = 0, \quad j = 1,...,c, \quad i = 1,...,m. \tag{25}$$

Multiplying both sides by $w_{i,j}^{-1}$, Eq. (25) becomes

$$0 = (p+1)w_{i,j}^{2(p+1)-1}w_{i,j}^{-1}(X_i - C_j)^2 + w_{i,j}w_{i,j}^{-1}\lambda_i = (p+1)w_{i,j}^{2p}(X_i - C_j)^2 + \lambda_i, \tag{26}$$

thus,

$$w_{i,j}^2 = k_{i,j;2} = \left(\frac{-\lambda_i}{(p+1)(X_i - C_j)^2}\right)^{1/p}. \tag{27}$$

Summing over cluster index j and applying constraint (17) we get

$$\sum_{j=1}^{c} w_{i,j}^2 = 1 = (-\lambda_i)^{1/p}\sum_{j=1}^{c}\left(\frac{1}{(p+1)(X_i - C_j)^2}\right)^{1/p}, \tag{28}$$

therefore

$$(-\lambda_i)^{1/p} = \frac{1}{\sum_{j=1}^{c}\left(\frac{1}{(p+1)(X_i - C_j)^2}\right)^{1/p}}. \tag{29}$$

Substitution of Eq. (29) into Eq. (27) leads to Eq. (21) where $d(\cdot) = ||\cdot||$ (Eucl. dist.). Note that if $p \to 0$, then Eq. (21) becomes equivalent to the WTA kernel function (15) and Eq. (22) reduces to Eq. (1). See Appendix 2 to examine how Eq. (21) relates to Eq. (5).

A soft-to-hard clustering algorithm whose kernel function always satisfies Eq. (5), whatever the value of the scale parameter may be, is Neural-Gas (see Appendix I in [3]). In other words, the cost function minimized by Neural-Gas is a special case of Eq. (2) for which Eqs. (3) to (12) hold true. This cost function is

$$E_{NG} = \sum_{i=1}^{m}\sum_{h=1}^{c}||X_i - C_h||^2 k_{i,h;3}, \tag{30}$$

such that

$$k_{i,h;3} = k_\lambda(r_h(X_i,\mathbf{C})) = e^{-r_h(X_i,\mathbf{C})/\lambda}, \tag{31}$$

where λ is a scale parameter monotonically decreasing with time and $r_h(X_i,\mathbf{C})$ is the neighborhood-ranking of vector C_h such that $r_h(X_i,\mathbf{C}) = 0$ if C_h is the best-matching unit, i.e., $C_h = C_{w1(i)}$, otherwise $r_h(X_i,\mathbf{C}) = 1$ if C_h is the second best-matching unit, etc. When $\lambda \to 0$, Eq. (30) becomes equivalent to Eq. (1).

Finally, let us briefly review SOM which differs from the previous learning frameworks by employing weighting values that are functions of internode distances in the output lattice rather than functions of interpattern distances in the input space. In deeper detail the SOM updating strategy differs from Eq. (7) as shown below

$$C_j^{(t+1)} = C_j^{(t)} + \epsilon(t)(X^{(t)} - C_j^{(t)})k_{\sigma(t)}(l(j,w1(t)), \quad j = 1,...,c, \tag{32}$$

where $w1(t)$ is the position, within the regular external lattice (graph) G, of the node featuring the largest activation value at presentation time t, $l(j, w1(t))$ is the lattice distance (neighborhood ranking in the external lattice) between lattice crossings j and $w1(t)$, and $k_{\sigma(t)}(l(j, w1(X(t)))$ is a unimodal function that decreases monotonically with increasing distance $l(j, w1(t))$ with a decay constant $\sigma(t)$, which is monotone-decreasing with t. For example,

$$k_{\sigma(t)}(l(j, w1(t)) = e^{-|j-w1(t)|^2/\sigma(t)^2}. \tag{33}$$

When $\sigma(t) \to 0$, $k_{\sigma(t)} \to 1$ iff $j = w1(t)$, and otherwise tends to zero, i.e., Eq. (33) becomes equivalent to Eq. (15) when $d(\cdot) = ||\cdot||$. Notice that SOM deals with topological relationships (adjacency, neighboring) among nodes in an output lattice G without dealing with internode connections (synaptic links) explicitly. It has been proved that, as long as Eq. (33) features $\sigma(t) > 0$ (i.e., as long as the adaptation strategy of SOM is soft competitive), one cannot specify a cost function that is minimized by Eq. (32), i.e., there exists no cost function yielding Eq. (32) as its stochastic gradient descent [3], [33], [34]. SOM instead features a set of potential functions, one for each node, to be independently minimized following a stochastic (on-line) gradient descent [34]. In [10], a cost function that leads to an update strategy similar to, but not precisely the same as, Eqs. (32) and (33) 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 [35].

In unsupervised learning algorithms featuring a firm statistical foundation, an example of soft-to-hard update strategy transition can be derived from EM parameter optimization of Gaussian mixtures. The optimization problem is expressed as the maximization of the joint probability of a dat set of m observed vectors $\mathbf{X} = (X_1, ..., X_m)$ (evidence), conditioned on a set of adjustable parameters $\theta = (\theta_1, ..., \theta_c)$. The probability density function $p(\mathbf{X}|\theta) = \mathcal{L}(\theta)$ represents the degree to which the data model fits the observed data, or it can be viewed as a function of θ for fixed $\mathbf{X}$, in which case it is referred to as the likelihood of parameter set θ for the given data set $\mathbf{X}$. Likelihood $p(\mathbf{X})$ can be represented as a mixture distribution, i.e., a linear combination of component densities $p(\mathbf{X}|j)$, $j = 1, ..., c$. According to the expansion rule [36], and under the hypothesis of i.i.d. sampling, we can write the negative log-likelihood for the data set as [2] [4], [28],

$$E_{ml} = -ln[p(\mathbf{X})] = -ln \prod_{i=1}^{m} p(X_i) = -\sum_{i=1}^{m} ln \sum_{j=1}^{c} p(X_i|j)p(j), \tag{34}$$

which can be regarded as an error function E_{ml} to be minimized, where ml stands for maximum likelihood. In Eq. (34), the mixing parameters $p(j)$, $j = 1, ..., c$, are called the *prior* probability of the data points generated from component j of the mixture [4]. These priors satisfy conditions

$$0 \leq p(j) \leq 1, \quad j = 1, ..., c, \tag{35}$$

$$\sum_{j=1}^{c} p(j) = 1. \tag{36}$$

When component density functions $p(X_i|j)$ are (isotropic) Gaussians, i.e.,

$$p(X_i|j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{|X_i - C_j|^2}{2\sigma_j^2}}, \quad \forall X_i \in \mathcal{R}^n, j = 1, ..., c. \tag{37}$$

then $p(\mathbf{X})$ is modeled as a Gaussian mixture. It is easy to prove that Maximum-Entropy cost function (19) is a special case of the negative log-likelihood of a Gaussian mixture: in particular, Eq. (19) is equivalent to the combination of Eqs. (34) and (37) where $\sigma_j = \sigma$, $j = 1, .., c$, and $p(j) = 1/c$, $j = 1, .., c$ (i.e., we have little prior information, or priors are the same and can be ignored [4], [18]). To maximize the combination of Eq. (34) with Eq. (37), i.e., to find the maximum likelihood solution of a Gaussian mixture, zero crossing of partial derivatives [4] (p. 62) and the EM optimization algorithm both provide a batch and iterative update rule for Gaussian centers which is [4] (p. 67)

$$C_j^{(e+1)} = \frac{\sum_{i=1}^{m} X_i p(j|X_i)}{\sum_{g=1}^{m} p(j|X_g)}, \quad j = 1, ..., c, \tag{38}$$

$$(\sigma_j^2)^{(e+1)} = \frac{1}{n} \frac{\sum_{i=1}^{m} p(j|X_i) ||X_i - C_j^{(e+1)}||^2}{\sum_{g=1}^{m} p(j|X_g)}, \quad j = 1, ..., C \tag{39}$$

$$p(j)^{(e+1)} = \frac{1}{m} \sum_{i=1}^{m} p(j|X_i), \tag{40}$$

where n is the dimensionality of the input space and $p(j|X_i)$ is the posterior probability of mixture component j given the observable vector X_i, defined according to Bayes' rule as

$$p(j|X_i) = \frac{p(X_i|j)p(j)^{(e)}}{\sum_{h=1}^{c} p(X_i|h)p(h)^{(e)}}. \tag{41}$$

Posterior probabilities satisfy conditions

$$0 \leq p(j|X_i) \leq 1, \quad \forall X_i \in \mathcal{R}^n, j = 1, ..., c, \tag{42}$$

$$\sum_{j=1}^{c} p(j|X_i) = 1, \quad \forall X_i \in \mathcal{R}^n. \tag{43}$$

Notice that Eq. (38) becomes equivalent to Eq. (6) by choosing

$$p(j|X_i) = k_j(d(X_i, \mathbf{C}^{(e)})),$$

where, on the basis of Eqs. (42) and (43), weighting values $k_j(d(X_i, \mathbf{C}^{(e)}))$, $\forall X_i \in \mathcal{R}^n$, $j = 1, ...c$, must satisfy Eqs. (16) and (17), i.e., the weighting function must be chosen as a relative fuzzy (i.e., useful [37]) membership function approximating a posterior probability.

By assuming that mixture components are spherical Gaussians having a common spread parameter, i.e., $\sigma_j = \sigma$, $j = 1, ..., c$, it can be proved that when $\sigma \to 0$, then the posterior probabilities for all mixture components is zero except for the mixture component $w1(i) \in \{1, c\}$ whose center vector $C_{w1(i)}$ is closest to X_i [4] (p. 190). This means that the EM update formula (38) applied to a mixture of spherical Gaussians featuring the same spread parameter $\sigma \to 0$ becomes equivalent to the batch update formula of the hard (purely competitive) c-means clustering problem (see Eq. (1) when $d(\cdot) = ||\cdot||$); this is tantamount to saying that the hard c-means clustering problem is a special case of EM maximization of the joint probability $p(\mathbf{X})$ modeled as a mixture of spherical Gaussians featuring the same spread parameter $\sigma \to 0$. In the literature, there are criticisms of this conclusion [2]

(p. 181): it can be shown that when the mean of a Gaussian is set to the value of any training data point, if $\sigma \rightarrow 0$ then the log-likelihood function increases indefinitely, i.e., no global maximum of the log-likelihood does exist, indicating that the maximum-likelihood approach fails to provide a definite solution. In other words, if the mean of a Gaussian is set to the value of any training data point, then there is no value of σ that gives a global maximum. Thus, the maximum likelihood inductive principle mail fail to provide a solution for estimation of simple densities such as the mixture of Gaussians. This calls into question the validity of considering Eqs. (38) and (39) when $\sigma \rightarrow 0$ as an applicable case.

3 Conclusions

Our first conclusion is that FLVQ, by adopting Eqs. (21) and (24), does not minimize cost function (22). This is made evident by considering that Eq. (24) assumes that Eq. (5) is always satisfied, while Eq. (5) holds true iff the FLVQ scale parameter p goes to zero (see Appendix 4). In other words, the cost function minimized by FLVQ is still unknown [38]. Experimentally, it is easy to verify that FLVQ iterations do not always reduce the cost given by the combination of Eqs. (21) and (22) when scale parameter p is fixed. For example, Figs. 1 and 2 show, respectively, the HCM and FLVQ clustering of a non-convex data set consisting of a circular ring plus three Gaussian clusters. Fig. 3 shows that when $p \leq 1.5$ is fixed, the FLVQ cost value, equivalent to the combination of Eqs. (21) and (22), increases. Figs. 2 and 4 show the typical behavior of a deterministic annealing procedure, such as FLVQ and Maximum-Entropy, where at a high initial temperature all reference vectors are attracted by the grand mean (center of gravity) of the data set [38].

Our second conclusion is that NG differs from FLVQ and the Maximum-Entropy clustering algorithm in satisfying Eq. (5), which is related to the update equation (6) and cost function (2), for every value of the scale parameter, while FLVQ and Maximum-Entropy satisfy Eq. (5) only in a limit case (when the scale parameter goes to zero). The consequence of this fact in terms of minimization of Eq. (1) is still unclear and requires further investigation because, despite the fact that NG, FLVQ and Maximum-Entropy all employ an update equation of type (6), each one of these algorithms starts from a different cost function that does not necessarily belong to type (2) (like Eq. (19), which is the cost function of Maximum-Entropy, while the cost function of FLVQ is unknown).

It is interesting to observe that an axiomatic learning framework, which is able to unify the expressions of the Maximum-Entropy and FLVQ cost functions, has been recently proposed [39]. In our further investigations we will try to find relationships between the unifying learning framework proposed in [39] and Eqs. (2), (4)-(6).

Appendix 1

For the sake of simplicity, let us consider a mapping between two 1-D spaces, where (x_h, y_h), $h = 1, ..., M$, $x_h \in \mathcal{R}$, $y_h \in \mathcal{R}$ is the supervised training data set. Instance-based learning approaches actually construct a different approximation to the target function for each distinct query instance [**?**]. Unobserved function values are estimated by fitting the nearby training points well, with less concern for distant training points according to the cost

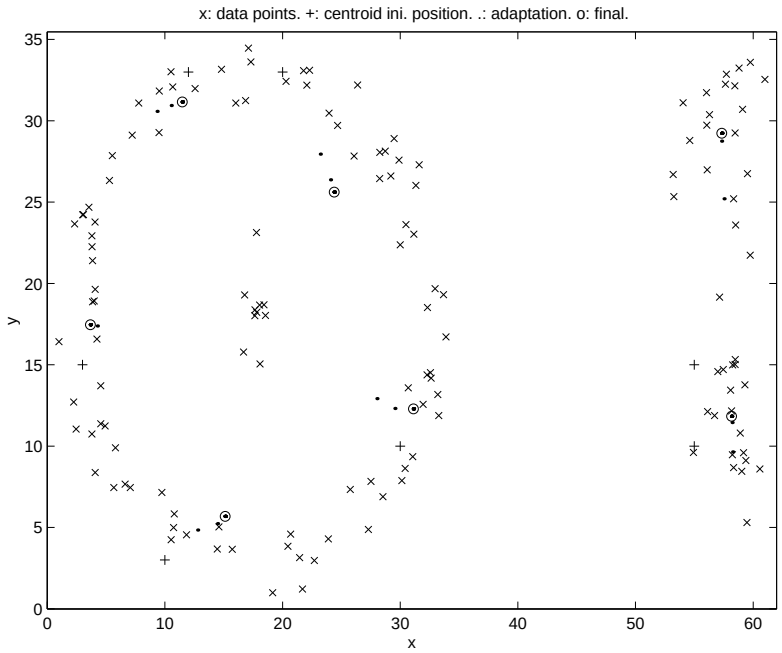

Figure 1: HCM clustering of a circular ring plus three Gaussian clusters: 140 data points; 7 clusters, 4 iterations. Final MSE = 30.92.

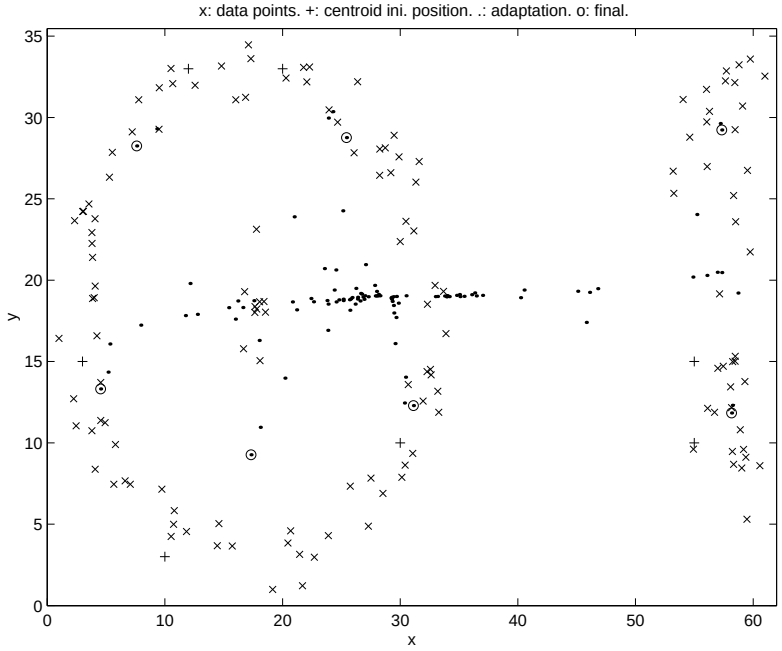

Figure 2: FLVQ clustering of a circular ring plus three Gaussian clusters: 140 data points; 7 clusters, 15 iterations. Final MSE = 30.80.

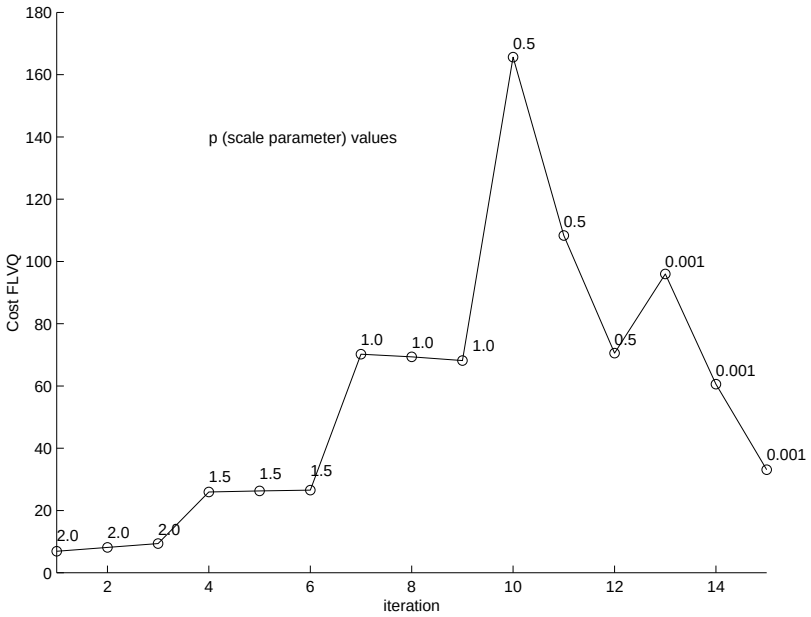

Figure 3: FLVQ cost function: if p is fixed and $p \geq 1.5$, the cost value increases at each iteration.

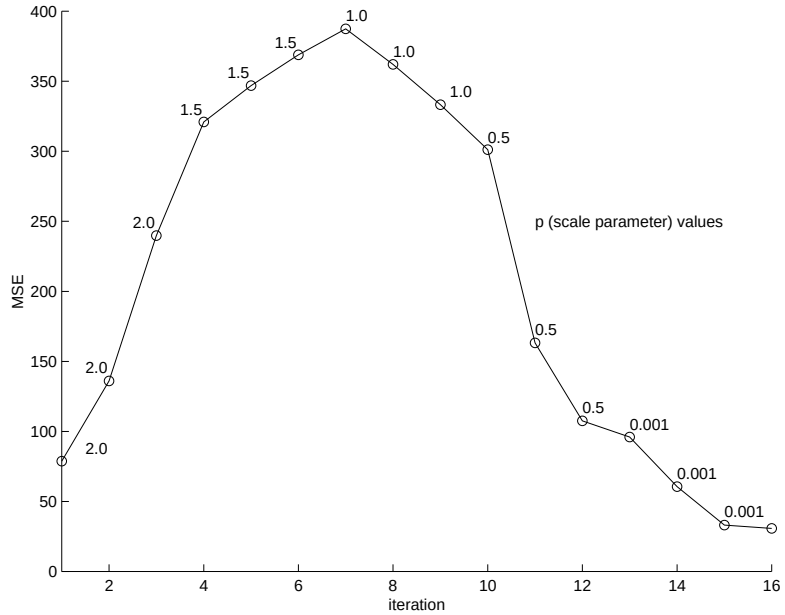

Figure 4: MSE values generated by the FLVQ clustering.

function

$$C(q) = \sum_{h=1}^{M} (\hat{y}(q) - y_h)^2 k(d(q, x_h)), \tag{A1.1}$$

where q is a query input point (unobserved example) and $k(d(q, x_h)) \geq 0$ is a distance-weighting function constrained by Eq. (XXX). The best estimate $\hat{y}(q)$ is the one that minimizes $C(q)$ such that

$$\frac{\partial C(q)}{\partial \hat{y}(q)} = \sum_{h=1}^{M} (\hat{y}(q) - y_h) k(d(q, x_h)) = 0,$$

therefore,

$$\hat{y}(q) = \frac{\sum_{h=1}^{M} y_h k(d(q, x_h))}{\sum_{h=1}^{M} k(d(q, x_h))}. \tag{A1.2}$$

According to Eq. (A1.2), if $k(d(q, x_h)) \to \infty$ when $d(q, x_h) \to 0$, i.e., when $q \to x_h$, then $\hat{y}(q) \to y_h$, i.e., exact interpolation of the supervised training data is pursued. If data are noisy, exact interpolation is not desirable, and Eq. (A1.2) should employ a weighting scheme based on finite values of the kernel function to guarantee smooth interpolation between training points [13]. For example, let us consider the case in which Eq. (A1.2) employs a Gaussian weighting function

$$k(d(q, x_h)) = e^{-\frac{(d(q, x_h))^2}{2\sigma_w}}. \tag{A1.3}$$

In this case, if $d(q, x_h) \to 0$, i.e., if $q \to x_h$, then $k(d(q, x_h)) \to 1$ and, as a consequence, $\hat{y}(q) \neq y_h$. This means that when Eq. (A1.2) employs a Gaussian weighting function then no exact interpolation is pursued because not every observed input value is mapped exactly onto its corresponding target value.

Appendix 2

Let us prove that when Eq. (15) employs $d(\cdot) = ||\cdot||$ (Euclidean distance), then terms R_j, $j = 1, ..., c$, vanish in Eq. (3). This demonstration derives from an analogous proof originally proposed in [3]. For notational convenience we use symbol $d_{i,j} = ||X_i - C_j||$ and $d_{i,\mathbf{C}} = d(X_i, \mathbf{C})$ in Eq. (4), which becomes

$$R_j = \sum_{i=1}^{m} \sum_{h=1}^{c} d_{i,h}^2 \frac{\partial k_h(d_{i,\mathbf{C}})}{\partial C_j} = \sum_{i=1}^{m} \left[d_{i,j}^2 \frac{\partial k_j(d_{i,\mathbf{C}})}{\partial C_j} + \sum_{h=1, h\neq j}^{c} d_{i,h}^2 \frac{\partial k_h(d_{i,\mathbf{C}})}{\partial C_j} \right], \quad j = 1, ..., c, \tag{A2.1}$$

where weighting function $k_h(d(X_i, \mathbf{C}))$ is computed according to Eq. (14), which can be written as

$$k_j(d(X_i, \mathbf{C})) = \theta_1 \left(\sum_{l=1, l\neq j}^{c} \theta_2(d_{i,j}^2 - d_{i,l}^2) \right), \tag{A2.2}$$

where $\theta_2(\cdot)$ is the Heaviside step function defined as

$$\theta_2(x) = \begin{cases} 1, \text{ if } x > 0, \\ 0, \text{ otherwise}, \end{cases} \tag{A2.3}$$

such that

$$\sum_{l=1,l\neq j}^{c} \theta_2(d_{i,j}^2 - d_{i,l}^2) \in \{0, ..., c-1\},$$

where the output zero condition occurs iff $d_{i,j} \leq d_{i,l}$, $l = 1, ..., c$.

In Eq. (A2.2), function $\theta_1(\cdot)$ is the Heaviside step function defined as

$$\theta_1(x) = \begin{cases} 1, \text{ if } x \leq 0, \\ 0, \text{ otherwise.} \end{cases} \quad (A2.4)$$

Notice that in Eq. (A2.4) the output one condition occurs iff $d_{i,j} \leq d_{i,l}$, $l = 1, ..., c$. It is known that the derivative of the Heaviside step function is such that $d\theta(y)/dy = \delta(y)$, where $\delta(y)$ is the delta distribution defined as

$$\delta(y) = 0, \text{if } x \neq 0,$$

and

$$\int \delta(y)dy = 1.$$

At this point we can compute the derivative in Eq. (4), which becomes

$$R_j = \sum_{i=1}^{m} d_{i,j}^2 \cdot 2 \cdot d_{i,j} \cdot \delta\left(\sum_{l=1,l\neq j}^{c} \theta_2(d_{i,j}^2 - d_{i,l}^2)\right) \sum_{l=1,l\neq j}^{c} \delta(d_{i,j}^2 - d_{i,l}^2)$$

$$+ \sum_{i=1}^{m} \sum_{h=1,h\neq j}^{c} d_{i,h}^2 \cdot (-2) \cdot d_{i,j} \cdot \delta\left(\sum_{l=1,l\neq j}^{c} \theta_2(d_{i,h}^2 - d_{i,l}^2)\right) \delta(d_{i,h}^2 - d_{i,j}^2). \quad (A2.5)$$

For every X_i, $i = 1, ..., c$, each of the c sums in the second term of Eq. (A2.5) is nonvanishing iff $d_{i,h} = d_{i,j}$. By removing all vanishing sums in this second term, Eq. (A2.5) becomes

$$R_j = \sum_{i=1}^{m} d_{i,j}^2 \cdot 2 \cdot d_{i,j} \cdot \delta\left(\sum_{l=1,l\neq j}^{c} \theta_2(d_{i,j}^2 - d_{i,l}^2)\right) \sum_{l=1,l\neq j}^{c} \delta(d_{i,j}^2 - d_{i,l}^2)$$

$$+ \sum_{i=1}^{m} d_{i,j}^2 \cdot (-2) \cdot d_{i,j} \cdot \delta\left(\sum_{l=1,l\neq j}^{c} \theta_2(d_{i,j}^2 - d_{i,l}^2)\right) \sum_{h=1,h\neq j}^{c} \delta(d_{i,h}^2 - d_{i,j}^2). \quad (A2.6)$$

Since $\delta(y) = \delta(-y)$, then R_j vanishes, $j = 1, ..., c$.

Appendix 3

If Eq. (18) is the kernel function adopted in Eq. (2), i.e.,

$$k_j(\sigma, d_{i,\mathbf{C}}) = \frac{e^{-\frac{d_{i,j}^2}{\sigma^2}}}{\sum_{g=1}^{c} e^{-\frac{d_{i,g}^2}{\sigma^2}}}, \quad j = 1, ..., c,$$

where $d_{i,j}^2 = ||X_i - C_j||^2 = (X_i - C_j)^T(X_i - C_j)$ and scale parameter $\sigma > 0$, then, according to Eq. (4), the term R_j becomes

$$R_j = \sum_{i=1}^{m}\sum_{h=1}^{c} d_{i,h}^2 \frac{\partial k_h(d_i, \mathbf{c})}{\partial C_j} = \sum_{i=1}^{m}\sum_{h=1}^{c} d_{i,h}^2 \left[\frac{\partial A_{i,h}}{\partial C_j}\frac{1}{D_i} + \frac{\partial D_i}{\partial C_j}\left(-\frac{A_{i,h}}{D_i^2}\right)\right], \tag{A3.1}$$

where
$\frac{\partial d_{i,j}^2}{\partial C_j} = -2(X_i - C_j)$,
$A_{i,j} = e^{-\frac{d_{i,j}^2}{\sigma^2}}$,
$D_i = \sum_{g=1}^{c} e^{-\frac{d_{i,g}^2}{\sigma^2}}$,
$\frac{\partial D_i}{\partial C_j} = \frac{\partial A_{i,j}}{\partial C_j} = A_{i,j}2(X_i - C_j)/\sigma^2$,
such that, if $t = 1/\sigma$, then

$$\lim_{\sigma\to 0}\frac{A_{i,j}}{D_i} = \lim_{t\to\infty}\frac{A_{i,j}}{D_i} = \begin{cases} o(e^{-\alpha t^2}), \text{ if } j \neq w1(i), \\ \\ 1 + o(e^{-\alpha t^2}), \text{ if } j = w1(i), \end{cases} \tag{A3.2}$$

where index $w1(i)$ identifies the best-matching unit, $\alpha > 0$ and $o(e^{-\alpha t^2})$ is an infinitesimal quantity that goes to zero as $e^{-\alpha t^2}$. Eq. (A3.1) becomes

$$R_j = \sum_{i=1}^{m} d_{i,j}^2 \frac{A_{i,j}}{C_j}\frac{1}{D_i} + \sum_{i=1}^{m}\sum_{h=1}^{c}\frac{\partial A_{i,j}}{\partial C_j}\left(-\frac{A_{i,h}}{D_i^2}\right)d_{i,h}^2$$

$$= \sum_{i=1}^{m}\frac{2A_{i,j}}{\sigma^2 D_i}(X_i - C_j)d_{i,j}^2 + \sum_{i=1}^{m}\sum_{h=1}^{c}\frac{2A_{i,j}}{\sigma^2 D_i^2}(X_i - C_j)d_{i,h}^2(-A_{i,h}). \tag{A3.3}$$

Substituting $t = 1/\sigma$ in Eq. (A3.3) we get

$$R_j = \sum_{i=1}^{m}\left[\frac{2t^2 A_{i,j}}{D_i}(X_i - C_j)\left(d_{i,j}^2 - \sum_{h=1}^{c}\frac{A_{i,h}}{D_i}d_{i,h}^2\right)\right]. \tag{A3.4}$$

Based on Eqs. (A3.2) and (A3.4) we obtain

$$\lim_{t\to\infty} R_j = \lim_{t\to\infty}\sum_{i=1}^{m}\left[\frac{2t^2 A_{i,j}}{D_i}(X_i - C_j)\left(d_{i,j}^2 - d_{i,w1(i)}^2 + o(e^{-\alpha t^2})\right)\right]$$

$$= \lim_{t\to\infty}\begin{cases} \sum_{i=1,\ldots,m; j=w1(i);}\left[2t^2(1 + o(e^{-\alpha t^2}))(X_i - C_j)o(e^{-\alpha t^2})\right], \\ \\ \sum_{i=1,\ldots,m; j\neq w1(i);}\left[2t^2 o(e^{-\alpha t^2})(X_i - C_j)\left(d_{i,j}^2 - d_{i,w1(i)}^2 + o(e^{-\alpha t^2})\right)\right], \end{cases}$$

$$\leq \lim_{t\to\infty}\begin{cases} \sum_{i=1,\ldots,m; j=w1(i);}\left|\left[2t^2(1 + o(e^{-\alpha t^2}))(X_i - C_j)o(e^{-\alpha t^2})\right]\right|, \\ \\ \sum_{i=1,\ldots,m; j\neq w1(i);}\left|\left[2t^2 o(e^{-\alpha t^2})(X_i - C_j)\left(M_1\right)\right]\right|, \end{cases}$$

$$\leq \lim_{t\to\infty} \begin{cases} \sum_{i=1,...,m; j=w1(i);} M_2 t^2 o(e^{-\alpha t^2}) = 0, \\ \\ \sum_{i=1,...,m; j\neq w1(i);} M_3 t^2 o(e^{-\alpha t^2}) = 0, \end{cases}$$

where M_1, M_2 and M_3 are finite quantities ≥ 0. To summarize, when the kernel function is chosen according to Eq. (18), where $d(\cdot) = |\cdot|$, and $\sigma \to 0$, then Eq. (19) becomes equivalent to Eq. (1), which is a special case of Eq. (2), and Eq. (5) holds true.

Appendix 4

If Eq. (21) is the kernel function adopted in Eq. (2), i.e.,

$$k_j(p, d_{i,\mathbf{C}}) = \frac{\frac{1}{d_{i,j}^{2/p}}}{\sum_{g=1}^{c} \frac{1}{d_{i,g}^{2/p}}}, \quad j = 1, ..., c,$$

where scale parameter $p > 0$, then, according to Eq. (4), the term R_j becomes

$$R_j = \sum_{i=1}^{m} \sum_{h=1}^{c} d_{i,h}^2 \frac{\partial k_h(d_{i,\mathbf{C}})}{\partial C_j} = \sum_{i=1}^{m} \sum_{h=1}^{c} d_{i,h}^2 \left[\frac{\partial A_{i,h}}{\partial C_j} \frac{1}{D_i} + \frac{\partial D_i}{\partial C_j} \left(-\frac{A_{i,h}}{D_i^2} \right) \right], \tag{A4.1}$$

where
$\frac{\partial d_{i,j}^2}{\partial C_j} = -2(X_i - C_j)$,
$A_{i,j} = \frac{1}{d_{i,j}^{2/p}}$,
$D_i = \sum_{g=1}^{c} e^{-\frac{d_{i,g}^2}{\sigma^2}}$,
$\frac{\partial D_i}{\partial C_j} = \frac{\partial A_{i,j}}{\partial C_j} = \frac{2}{p} \frac{A_{i,j}}{(X_i - C_j)}$,
such that, if $t = 1/p$, then

$$\lim_{p\to 0} \frac{A_{i,j}}{D_i} = \lim_{t\to\infty} \frac{A_{i,j}}{D_i} = \begin{cases} O(\theta^t), \text{ if } j \neq w1(i), \\ 1 + O(\theta^t), \text{ if } j = w1(i), \end{cases} \tag{A4.2}$$

where index $w1(i)$ identifies the best-matching unit, $0 \leq \theta < 1$ and $O(\theta^t)$ is an infinitesimal quantity that goes to zero as θ^t.

By analogy with Eq. (A3.4), it is possible to prove that

$$R_j = \sum_{i=1}^{m} \left[\frac{2tA_{i,j}}{D_i(X_i - C_j)} \left(d_{i,j}^2 - \sum_{h=1}^{c} \frac{A_{i,h}}{D_i} d_{i,h}^2 \right) \right]. \tag{A4.3}$$

Based on Eqs. (A4.2) and (A4.3) we obtain

$$\lim_{t\to\infty} R_j = \lim_{t\to\infty} \sum_{i=1}^{m} \left[\frac{2tA_{i,j}}{D_i(X_i - C_j)} \left(d_{i,j}^2 - d_{i,w1(i)}^2 + O(\theta^t) \right) \right]$$

$$= \lim_{t\to\infty} \begin{cases} \sum_{i=1,...,m; j=w1(i);} \left[2t(1 + O(\theta^t)) \frac{1}{(X_i - C_j)} O(\theta^t) \right], \\ \\ \sum_{i=1,...,m; j\neq w1(i);} \left[2tO(\theta^t) \frac{1}{(X_i - C_j)} \left(d_{i,j}^2 - d_{i,w1(i)}^2 + O(\theta^t) \right) \right], \end{cases}$$

$$\leq \lim_{t \to \infty} \begin{cases} \sum_{i=1,\ldots,m; j=w1(i);} M_1 tO(\theta^t) = 0, \\ \\ \sum_{i=1,\ldots,m; j \neq w1(i);} M_2 tO(\theta^t) = 0, \end{cases}$$

where M_1 are M_2 are finite quantities ≥ 0. To summarize, when the kernel function is chosen according to Eq. (21), where $d(\cdot) = |\cdot|$, and $p \to 0$, then Eq. (22) becomes equivalent to Eq. (1), which is a special case of Eq. (2), and Eq. (5) holds true.

References

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

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

[3] 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.

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

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

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

[7] S. M. Omohundro, "The Delaunay triangulation and function learning," International Computer Science Institute, Berkeley, CA, TR-90-001, 1990.

[8] J. R. Shewchuck, "Delaunay refinement mesh generation," Carnegie Mellon University, Pittsburgh, PA, CMU-CS-97-137, 1997.

[9] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

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

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

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

[13] C. G. Atkeson, S. A. Schall, and A. W. Moore, "Locally weighted learning," *AI Review*, vol. 11, pp. 11-73, 1997.

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

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

[16] 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.

[17] 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.

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

[19] 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.

[20] L. Bottou, "Online learning and stochastic approximations," in *Online Learning in Neural Networks*, D. Saad, Ed., Cambridge: Cambridge University Press, 1998.

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

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

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

[24] T. Hofmann and J. M. Buhmann, "Competitive learning algorithms for robust vector quantization," *IEEE Trans. on Signal Processing*, vol. 46, no. 6, pp. 1665-1675, 1998.

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

[26] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition," International Computer Science Institute, Berkeley, CA, TR-98-038.

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

[28] 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.

[29] 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.

[30] A. Baraldi and F. Parmiggiani, "A fuzzy neural network model capable of generating/removing neurons and synaptic links dynamically," *Proc. WILF '97 - II Italian Workshop on Fuzzy Logic,* P. Blonda, M. Castellano and A. Petrosino, Eds., World Scientific, Singapore, 1998, pp. 247-259.

[31] 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.

[32] J. C. Bezdek,"A convergence theorem for the fuzzy ISODATA clustering algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, no. 1, pp. 1-8, 1980.

[33] 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.

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

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

[36] D. Heckerman, "A tutorial on learning with Bayesian networks," Microsoft Technical Report, MSR-TR-95-06, 1995.

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

[38] 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, pp. 724-738, 1998.

[39] N. B. Karayiannis, "Reformulating learning vector quantization and radial basis function networks," *Fundamenta Informaticae*, vol. 37, pp. 137-175, 1999.