SELF-ORGANIZATION AND CLUSTERING ALGORITHMS

James C. Bezdek
Div. of Computer Science
University of West Florida
Pensacola, Florida 32514

This research was partially supported by NSF Grant # IRI-9003252

ABSTRACT

Kohonen's "feature maps" approach to clustering is often likened to the k or c-means clustering algorithms. In this note we identify some similarities and differences between the hard and fuzzy c-Means (HCM/FCM) or ISODATA algorithms and Kohonen's "self-organizing" (KSO) approach. We conclude that some differences are significant, but at the same time there may be some important unknown relationship(s) between the two methodologies. We propose several avenues of research which, if successfully resolved, would strengthen both the HCM/FCM and Kohonen clustering models. We do not, in this note, address aspects of the KSO method related to associative memory and to the feature map display technique.

1. INTRODUCTION

Treatments of many classical approaches to clustering appear in Kohonen [1], Bezdek [2], and Duda and Hart [3]. Kohonen's work has become particularly timely in recent years because of the widespread resurgence of interest in Artificial Neural Network (ANN) structures. ANNs and pattern recognition are discussed by Pao [4] and Lippman [5]. Our interest lies with the KSO algorithm as it relates to the solution of clustering and classification problems and the HCM/FCM models.

2. CLUSTERING ALGORITHMS AND CLASSIFIER DESIGN

Let (c) be an integer, $1< c < n$ and let $X = \{x_1, x_2, ..., x_n\}$ denote a set of (n) feature vectors in $\Re^s$. X is *numerical object data*; the j-th object (some physical entity such as a medical patient, seismic record etc.) has vector x_j as it's numerical representation; x_{jk} is the k-th characteristic (or *feature*) associated with object j. Given X, we say that (c) fuzzy subsets $\{u_i: X \rightarrow [0,1]\}$ are a fuzzy c-partition of X in case the (cn) values $\{u_{ik} = u_i(x_k),\ 1 \le k \le n,\ 1 \le i \le c\}$ satisfy three conditions:

$$0 \le u_{ik} \le 1 \text{ for all } i,k \qquad (1a)$$

$$\Sigma u_{ik} = 1 \text{ for all } k \quad ; \tag{1b}$$

$$0 < \Sigma u_{ik} < n \text{ for all } i \quad . \tag{1c}$$

Each set of (cn) values satisfying conditions (1) can be arrayed as a (cxn) matrix $U = [u_{ik}]$. The set of all such matrices are the *non-degenerate fuzzy c-partitions* of X:

$$M_{fcn} = \{U \text{ in } \Re^{cn} \mid u_{ik} \text{ satisfies (1) for all i and k}\}. \tag{2}$$

And in case all the u_{ik}'s are either 0 or 1, we have the subset of *hard (or crisp) c-partitions* of X:

$$M_{cn} = \{U \text{ in } M_{fcn} \mid u_{ik} = 0 \text{ or } 1 \text{ for all i and k}\}. \tag{3}$$

The reason these matrices are called partitions follows from the interpretation of u_{ik} as the *membership* of $\mathbf{x}_k$ in the i-th partitioning subset (cluster) of X. M_{fcn} is more realistic as a physical model than M_{cn}, for it is common experience that the boundaries between many classes of real objects are in fact very badly delineated (i.e., really fuzzy). The important point is that *all* clustering algorithms generate solutions to the clustering problem for X which are matrices in M_{fcn}. The *clustering problem for X*, is, quite simply, the identification of an "optimal" partition U of X in M_{fcn}; that is, one that groups together object data vectors (and hence the objects they represent) which share some well defined (mathematical) similarity. It is our hope and implicit belief, of course, that an optimal mathematical grouping is in some sense an accurate portrayal of natural groupings in the physical process from whence the object data are derived. The number of clusters (c) must be known, or becomes an integral part of the problem.

3. THE ISODATA AND KSO ALGORITHMS

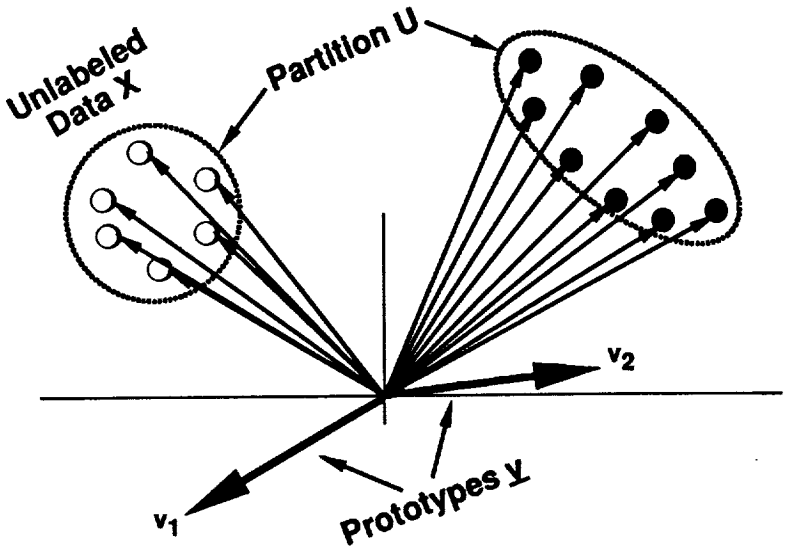
The most well known objective function for clustering is the least total squared error function:

$$J_1(U,\mathbf{v}\,;X) = \Sigma\Sigma\, u_{ik}\,(\|\mathbf{x}_k - \mathbf{v}_i\|_I)^2 \quad , \tag{4}$$

where $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c)$ is a vector of (unknown) cluster centers (weights or prototypes), $\mathbf{v}_i \in \Re^s$ for $1 \le i \le c$, $U \in M_{cn}$ is an unknown hard c-partition of X, and $\|\cdot\|_I$ is the Euclidean norm on $\Re^s$. Optimal partitions U^* of X are taken from pairs $(U^*, \mathbf{v}^*)$ that are "local minimizers" of J_1. It is important to recognize the geometric impact that the use of a norm function in J_1 as the criterion of (dis)similarity has on "good clusters" (here $\|\cdot\|_I$, but more generally, any norm on $\Re^s$ induced by a positive definite weight matrix A, as described below). Figure 1 illustrates this graphically; partitions that optimize J_1 will, generally speaking, contain clusters that conform to

the topology that is induced on $\mathfrak{R}^S$ by the eigenstructure of the norm-inducing matrix A. When A = I, good clusters will be hyperspherical, as the one in the left portion of Figure 1; otherwise, they will be hyperelliptical, as the one on the right side of Figure 1.

Figure 1. Geometry of Cluster Formation in Norm-Driven Clustering Algorithms



As is evident in Figure 1, clusters that optimize J_1 are formed on the basis of two properties: *location* and shape. Location information is contained in the lengths of the data vectors and "cluster centers" or prototypes $\{v_i\}$ from the origin, whilst shape information is embedded in the topology induced by the norm in use. Roughly speaking, these correspond to the mean and variance of probability distributions, so (4) is in some sense analogous to regarding the data as being drawn from a mixture of probability density functions (indeed, there are special cases when (4) yields identical results to the maximum likelihood estimators of the parameters of a mixture of normal distributions). Although the norm shown in (4) is the Euclidean norm, generalizations of J_1 have used all five of the usual norms encountered in numerical analysis and pattern recognition - viz, the Euclidean, Diagonal and Mahalonobis inner product A-norms; and the $p = 1$ and $p = \infty$ (city block and sup) Minkowski norms. The defining equations and unit ball shapes for these two families of norms are shown in Figure 2.

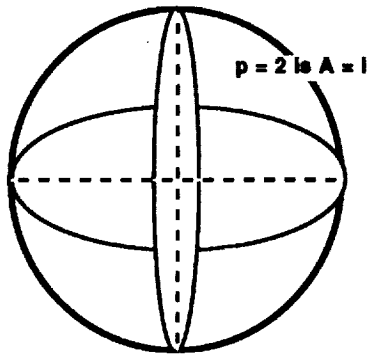
As an explicit means for finding optimal partitions of object data, J_1 was popularized as part of the ISODATA ("Iterative Self-Organizing Data Analysis") algorithm (c-Means + Heuristics) by Ball and Hall [6] in 1967. It is interesting to note that Kohonen apparently first used the term "self-organizing" to describe his approach

about 15 years later [1]. Apparently, the feature of both algorithms that suggests this phrase is their ability to iteratively adjust the weight vectors or prototypes that subsequently represent the data in an orderly and improving manner as the algorithms proceed with iteration. We contend that this use of the term "self-organizing" in the current context of neural network research is somewhat misleading (in both cases). Indeed, if the aspect of FCM/HCM and KSO that entitles us to call them self-organizing is their ability to adjust their parameters during "training", then every iterative method that produces approximations from data is self-organizing (e.g., Newton's method!). On the other hand, if this term serves to indicate that the algorithms in question can find meaningful labels for objects, without external interference (labelled) training examples), then all clustering algorithms are "self-organizing". Since the terminology in both cases is well established, the only expectation this writer has about the efficacy of these remarks is that they caution readers take the semantics associated with much of the current Neural Network literature with a large grain of salt.

Figure 2. Geometry of Level Sets for Inner product A-norms and Minkowski p-norms

Unit Ball Shapes in the A - norms

$L_A = \{x : \langle x,x \rangle_A = x^T Ax = (\|x\|_A)^2 = 1\}$



$$\| x_k - v_i \|_A = ((x_k - v_i)^T A (x_k - v_i))^{(1/2)}$$

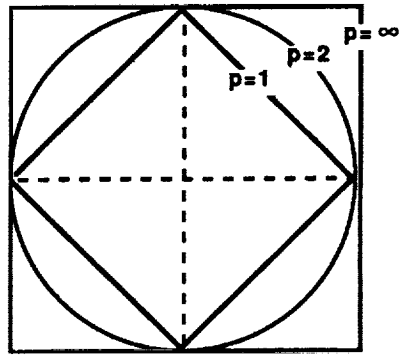
EV's of pos-definite (A) induce shapes

Inner product : Hilbert Space Structure

Differentiable in All Variables

Unit Ball Shapes in the p - norms

$L_p = \{x : \|x\|_p = 1\}$



$$\| x_k - v_i \|_p = (\Sigma \, |x_{kj} - v_{ij}|^p)^{(1/p)}$$

$$\| x_k - v_i \|_1 = (\Sigma \, |x_{kj} - v_{ij}|)$$

$$\| x_k - v_i \|_\infty = (\max_j \{|x_{kj} - v_{ij}|\})$$

p = 2 : Hilbert ; p ≠ 2: Banach Spaces

Dunn [7] first generalized J_1 by allowing U to be fuzzy (m=2 below) and the norm to be an arbitrary inner product A-norm. Bezdek [8] generalized Dunn's functional to the fuzzy ISODATA family written as:

$$J_m(U,v;X) = \Sigma\Sigma u_{ik}^{\ m}(||x_k-v_i||_A)^2, \tag{5}$$

where $m \in [1, \infty)$ is a weighting exponent on each fuzzy membership; $U \in M_{fcn}$ is a fuzzy c-partition of X; $\mathbf{v} = (v_1, v_2, ..., v_c)$ are cluster centers in $\Re^s$; **A** = is any positive definite (s x s) matrix; and $(||x_k-v_i||_A)^2 = (x_k-v_i)^T A (x_k-v_i)$ is the OG distance (in the A norm) from x_k to v_i .

In 1979, Gustafson and Kessel [8] derived necessary conditions to minimize an extension of (5) with (c) different norm inducing matrices. In 1981 Bezdek et. al. [9] generalized (5) by allowing the prototypes to be (convex combinations of) linear manifolds of arbitrary and different dimensions. In 1985 Pedrycz [10] introduced a way to use partially labeled data with (5) that amounts to a mixed supervised-unsupervised clustering scheme. In 1989 Dave [11] introduced a generalization of (5) that uses *hyperspherical* prototypes for **v**. In 1990 Bobrowski and Bezdek [12] used the city block and sup norms with (5), thus extending the c-Means algorithms to the most important Minkowski norms (p = 1 and p = ∞).

Necessary conditions that define iterative algorithms for (approximately) minimizing J_m and its generalizations are known. Our interest lies with the cases represented by (4) and (5). The conditions that are necessary for minima of J_1 and J_m follow :

<u>Hard c-Means (HCM) Theorem [2].</u> (U, **v**) may minimize $\Sigma\Sigma\, u_{ik}(||x_k- v_i||_A)^2$ only if

$$u_{ik} = \quad 1;\ (||x_k- v_i||_A)^2 = \min_j\{(||x_k- v_j||_A)^2\};\quad \text{and } =0;\ \text{otherwise} \tag{6a}$$

$$v_i = \Sigma u_{ik}x_k / \Sigma u_{ik} \tag{6b}$$

Note that HCM produces hard clusters $U \in M_{cn}$. The HCM conditions are necessary for "minima"of (4) (i.e., with A=I, the Euclidean norm on $\Re^s$), and, as we shall note, are also used to derive hard clusters in the KSO algorithm. The well known generalization of the HCM conditions is contained in the:

<u>Fuzzy c-Means (FCM) Theorem [2].</u> (U,**v**) may minimize $\Sigma\Sigma\, u_{ik}^{\ m}(||x_k- v_i||_A)^2$ for m > 1 only if :

$$u_{ik} = (\Sigma(||x_k- v_i||_A / ||x_k- v_j||_A)^{2/(m-1)})^{-1} \tag{7a}$$

$$v_i = \Sigma (u_{ik})^m x_k / \Sigma (u_{ik})^m \qquad (7b)$$

The FCM conditions are necessary for minima of (5). There is an alternative equation for (7a) if one or more of the denominators in (7a) is zero. These equations converge to the HCM equations as $m \to 1$ from above, and for $(m > 1)$, the U in FCM is truly fuzzy, i.e., $U \in (M_{fcn} - M_{cn})$. The FCM algorithms are simple Picard iteration through the paired variables U and **v**. Because we want to compare this method to the KSO algorithm, we give a brief description of the FCM/HCM algorithms.

(Parallel) c-Means (FCM/HCM) Algorithms

<FCM/HCM 1> : Given unlabeled data set $X = \{x_1, x_2, ..., x_n\}$. Fix : $1 < c < n$; $1 \le m < \infty$ (m=1 for HCM); positive definite weight matrix A to induce an inner product norm on $\Re^s$; and ε, a small positive constant.

<FCM/HCM 2>: Guess $v_0 = (v_{1,0}, v_{2,0}, ..., v_{c,0}) \in \Re^{cs}$ (or, initialize $U_0 \in M_{fcn}$).

<FCM/HCM 3>: For j = 1 to J:

<3a> : Calculate U_j with $\{v_{i,j-1}\}$;

<3b>: Update $\{v_{i,j-1}\}$ to $\{v_{i,j}\}$ with U_j ;

<3c>: If $\max_i \{ \|v_{i,j-1} \text{ to } v_{i,j}\| \} \le \varepsilon$, then stop and put $(U^*,v^*) = (U_j,v_j)$; Else : Next j

This procedure is known to converge q-linearly from any initialization to a local minimum or saddle point (U^*,v^*) of J_m. Note again that the update rule for the weights $\{v_i\}$ at step <3b> is a necessary condition for minimizing J_m. Moreover, all (c) weight vectors are updated using all (n) data points simultaneously at each pass; i.e., the weights $\{v_i\}$ are not sequentially updated as each x_k is processed. This is why we call the above description a "parallel" version of c-means, as opposed to the well known sequential version.

There is a *sequential* version of hard c-means (SHCM) that can be used to minimize J_1, and readers should be aware that it may produce quite different results than HCM on the same data set. One iteration of the SHCM algorithm is as follows: beginning with some hard U, the centers $\{v_i\}$ are calculated with (6b). Once the prototypes are known, one returns to update U. Beginning with x_1, each point is examined, and moved from,

say, cluster i to cluster j, so as to maximize the decrease in J_1 (if possible) . Then the two affected centers $\{v_i, v_j\}$ and rows i and j of U are updated using equations (6) . One complete pass of SHCM consists of testing each of the n data points in X, and effecting a transfer at each point where a decrease in J_1 can be realized. SHCM terminates when a complete pass can be made without transfers. We mention this version of HCM because it is SHCM that most closely resembles the KSO algorithm. Figure 3 is a rough depiction of how the HCM method might begin; Figure 4 indicates a desirable situation at termination. In Figure 3 the initial hard clusters subdivide the data badly, and the overall mean squared error (the sum of squares of the solid line distances between data points and prototypes) is large; at termination, the prototypes lie "centered" in their clusters, the overall sum of squared errors is low, and the hard 2-partition subdivides the data "correctly" (this is what happens if we are lucky !).

Figure 3. An Initial 2-Partition and Prototypes for HCM

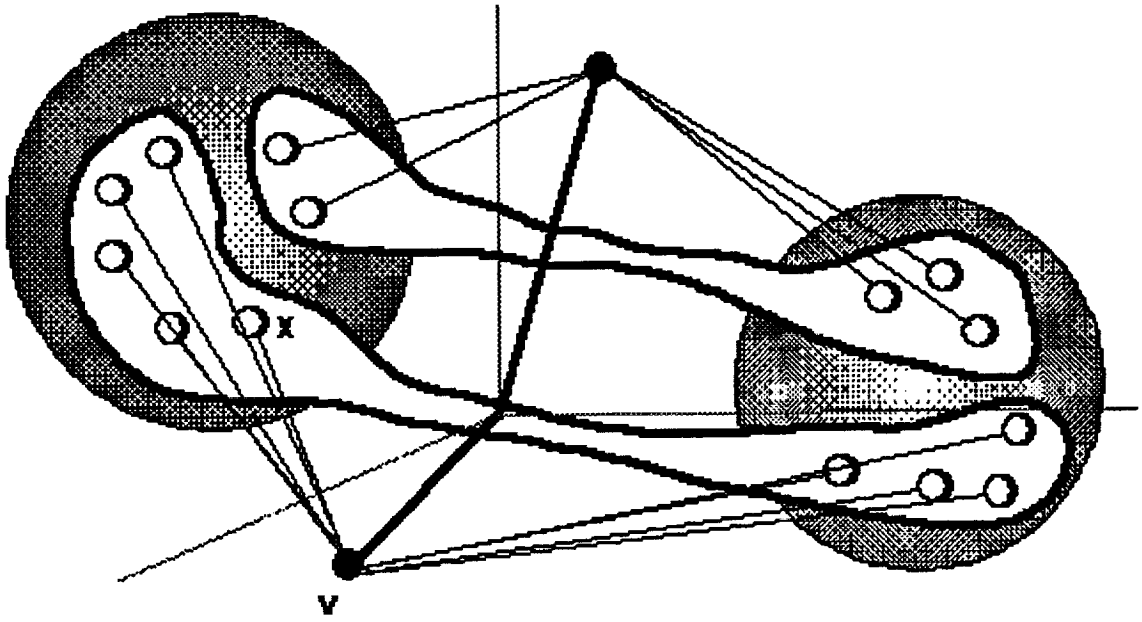
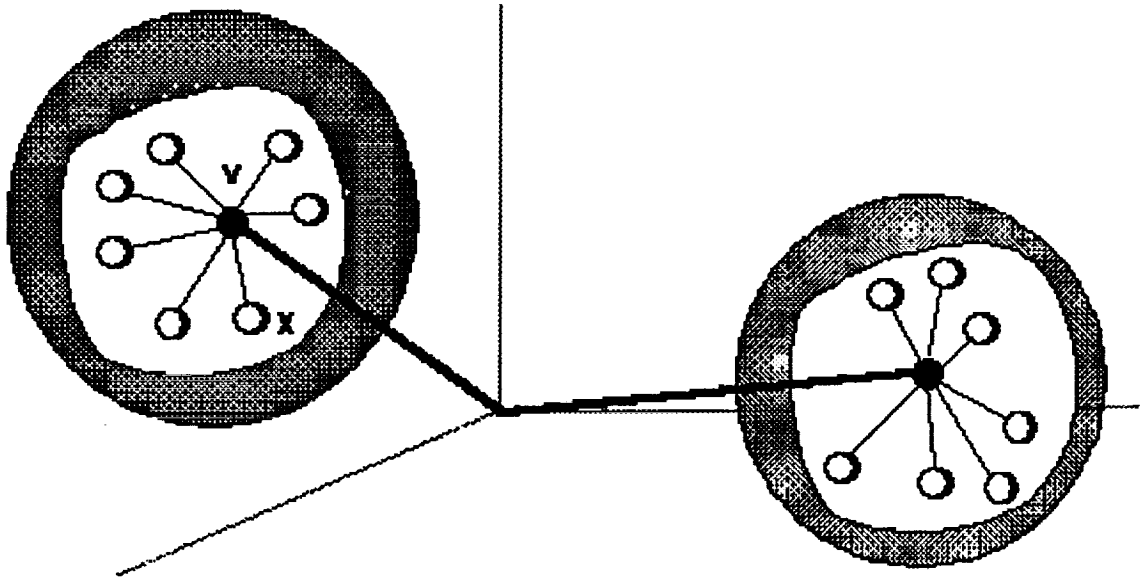


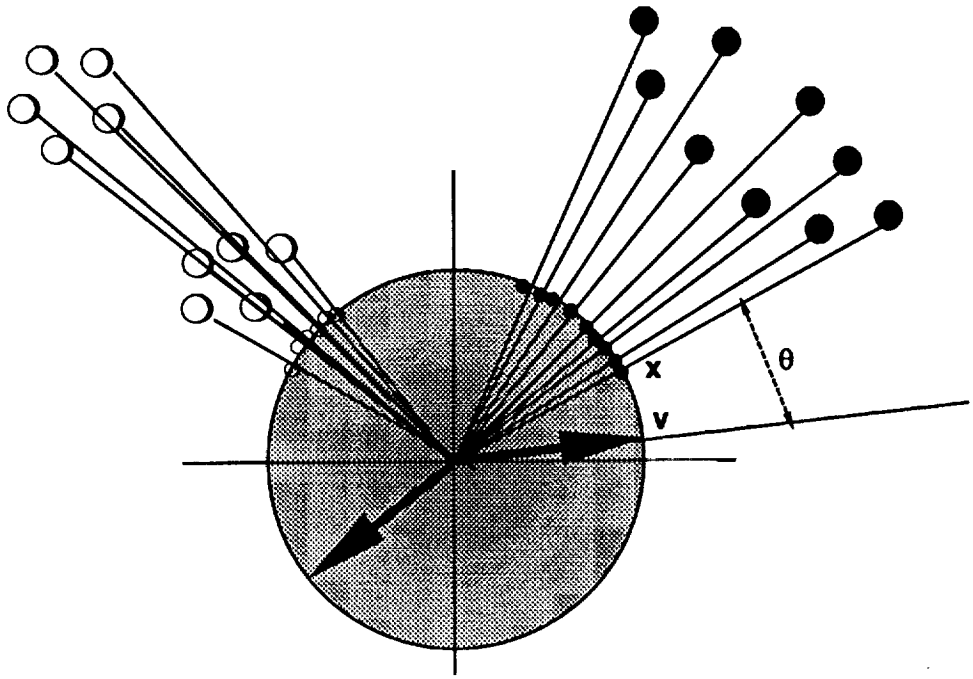
Figure 4. A (Benevolent) Final Configuration of 2-Partition and Prototypes for HCM



Kohonen's method differs from the c-means approach in several important ways. First, it is not a norm-driven scheme. Instead, the KSO method uses the geometric notion of *orientation matching*, depicted in Figure 5, as the basic measure of similarity between data points and cluster centers. Second, there is no partition U involved in the KSO algorithm. Instead, an initial set of cluster centers are iteratively updated without reference to partitions of the unlabeled data. The underlying geometry of the criterion of similarity is shown in Figure 5.

The measure of similarity, as shown in Figure 5, is the angle between a data point **x** and prototype **v** (in the neural network community, the vectors $\{\mathbf{v}_i\}$ are often called "weight" vectors; each one being attached or identified with a "node" in the network). Information that the data set may contain about cluster shapes in feature space is lost (i.e., not used by $\cos(\theta)$); and if the data are normalized at each step to be vectors of length 1, as they usually are in the KSO approach, location information is lost as well. Consequently, the geometry favored by the KSO criterion of similarity is data substructures that lie in *angular cones* emanating from the origin. We emphasize that in real data, either type of criterion - the c-means type norm driven measure, or the KSO angular measure - may or may not be appropriate for matching the data. As with all clustering problems, the question is not - which is better? the question is, which is better for this data set? In order to effect comparison with the c-means model, a brief description of Kohonen's algorithm follows.

Figure 5. Geometry of Cluster Formation in Orientation Matching Clustering Algorithms



$$\cos(\theta) = \langle x,v \rangle = 1 - \|x-v\|^2 / 2$$

Kohonen's (KSO) Clustering Algorithm

<KSO1> : Given unlabeled , "ordered" data set $X = \{x_1, x_2, ..., x_n\}$. Fix : $1 < c^*$; Choose update scale factors $\{\alpha_j\}$ so that $\{\alpha_j\} \to 0$; $\Sigma\alpha_j = \infty$, $\Sigma(\alpha_j)^2 < \infty$; Choose update neighborhood "radii" $\{\rho_j\} \in \{0,1,2,..., c^*\}$:

<KSO2> Guess (unit vectors) $v_0 = (v_{1,0}, v_{2,0}, ..., v_{c^*,0}) \in \Re^{c^*s}$

<KSO3> For j = 1 to J: : For k = 1 to n:

<3a> : Find i*(k) st $(\|x_k - v_{i^*(k)}\|_I)^2 = \min\{(\|x_k - v_{j(k)}\|_I)^2$

<3b>: For indices $N^*(k) = i^*(k), i^*(k) \pm 1, \ldots i^*(k) \pm \rho_j$ Update $v_{t,j-1}$:

$$v_{t,j} = v_{t,j-1} + \alpha_j (x_k - v_t) / \| v_{t,j-1} + \alpha_j (x_k - v_t) \|_I ; \text{ otherwise, } v_{t,j} = v_{t,j-1} . \quad (8)$$

Next k; Next j

We have used c^* instead of c in this procedure to emphasize the fact that Kohonen's method often uses "multiple" prototypes, in the sense that even though (unbeknownst to us !) X contains only c clusters, it may be advantageous to look for $c^* > c$ cluster centers; this is a further difference between the c-means and KSO strategies. This is one form of Kohonen's approach; other update rules have been used. The geometry of the update rule for the weight vectors in (8) is depicted in Figure 6. Thus, if we are at point x_k, as shown in Figure 6, <3a> of the KSO algorithm simply finds the current prototype (v_{old}) closest to x_k in angle (minimizing the angle is equivalent to the formula in <3a>). If the current center is called $v_{old} = v_{i^*(k)}$ as in Figure 6, then update equation (8) connects $v_{old} = v_{i^*(k)}$ to the vector x_k, rotates v_{old} to the new position v_{new}, and finally normalizes v_{new}.

The KSO procedure is exactly like SHCM in that it updates (some subset of the) prototypes sequentially after the examination of each data point. Figure 7 indicates the geometry of the scheme specified in <3b>; the basic idea is that once the prototype v_{old} closest to the current data point is found, all prototypes in a neighborhood of the "winner" are also updated.

Figure 6. The Geometry of Kohonen's Updating Rule

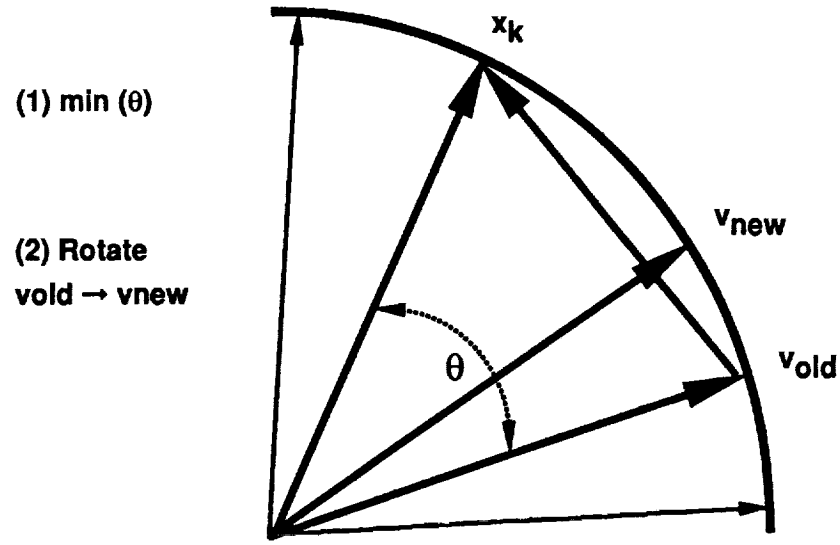
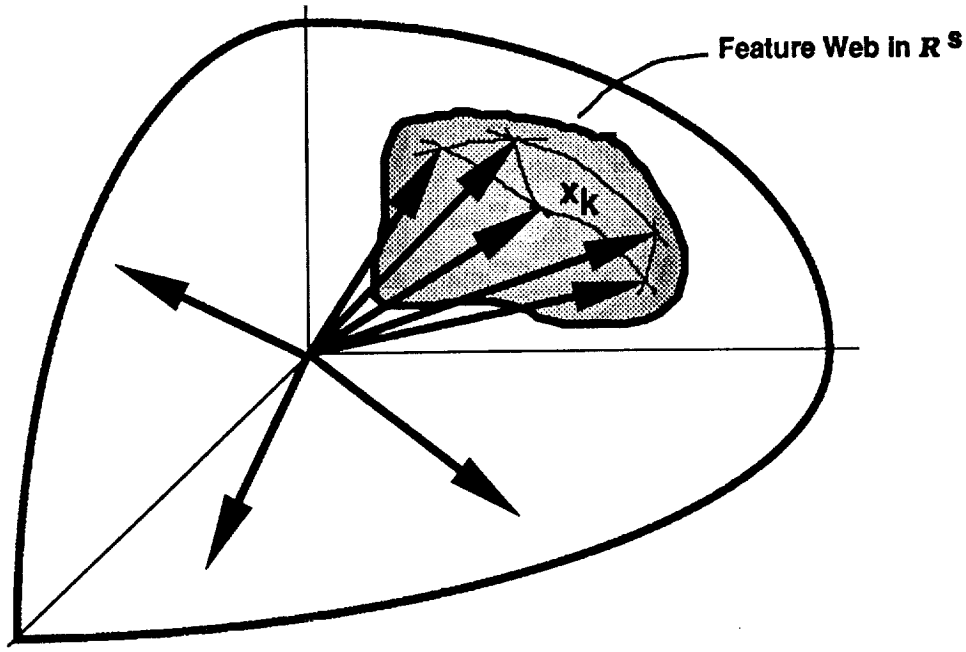


Figure 7. KSO Updating of Prototypes in the Neighborhood $N^*(k)$ of "Winner" $v_{l^*(k)}$



Although the "feature web" shown in Figure 7 is conceptualized here as being in $\mathfrak{R}^s$, it has actually been displayed only the case s = 2. Kohonen has shown that this process converges, in the sense that the $\{v_{t,j}\} \rightarrow \{v_t^*\}$ as $\{\alpha_j\} \rightarrow 0$, in the special case s=2. Moreover, the limiting $\{v_t^*\}$ preserve a "topological ordering" property of the data set X on an array of output nodes associated to the weight vectors. Iteration in the KSO method thus trains the weight vectors $\{v_t^*\}$ so that they preserve "order" in the output nodes. As previously noted, the KSO method does not use or generate a partition U of the data during training. However, once the weight vectors stabilize, the KSO model produces a hard U by following the nearest prototype rule below.

More specifically, once a set of prototypes $\{v_i\}$ are found by "training" on some data set X (this includes all four methods described above, HCM, FCM SHCM and KSO), they can be used to label *any* unlabeled data set. For *any* vector $\mathbf{x} \in \mathfrak{R}^s$, the HCM equation for u_{ik} defines a (piecewise linear) *nearest prototype classifier:*

The Nearest Prototype Classifier Decision Rule : Given $\{v_i\}$, Compute, *non-iteratively*, the hard c-partition of (*any*) data X with HCM equation (6a):

$$u_{ik} = \begin{cases} 1; & ((\|x_k - v_i\|_I))^2 = \min_j\{(\|x_k - v_j\|_I)^2\} \\ 0; & \text{otherwise} \end{cases} \qquad (9)$$

Note that we have written (9) with the Euclidean norm. Theorem 2 suggests that any scalar product induced A-norm might be used in the formula; however, interpretation of the subsequent decision rule as discussed above becomes very difficult. Thus, while it makes sense geometrically to consider variations in the norm as in (7) while *searching* for the cluster centers, it is much less clear that norms other than the Euclidean norm should be used during *classification*. Figure 8 is a rough depiction of how the KSO method might begin; Figure 9 shows the situation after termination of KSO, followed by *a posteriori* application of (9) to find an "optimal" hard c-partition U corresponding to the final weights . A question about how rule (9) is used with the KSO prototypes remains: how do we, without labeled data, assign one of $c < c^*$ "real" labels to subsets of the c^* weight vectors found by the KSO scheme? The same question applies to FCM - we still need to decide which of the c "real" labels belongs to each prototype - the problem is just more pronounced when there are multiple prototypes for each class.

Figure 8. Initial Configuration of Weight Vectors in the KSO Scheme

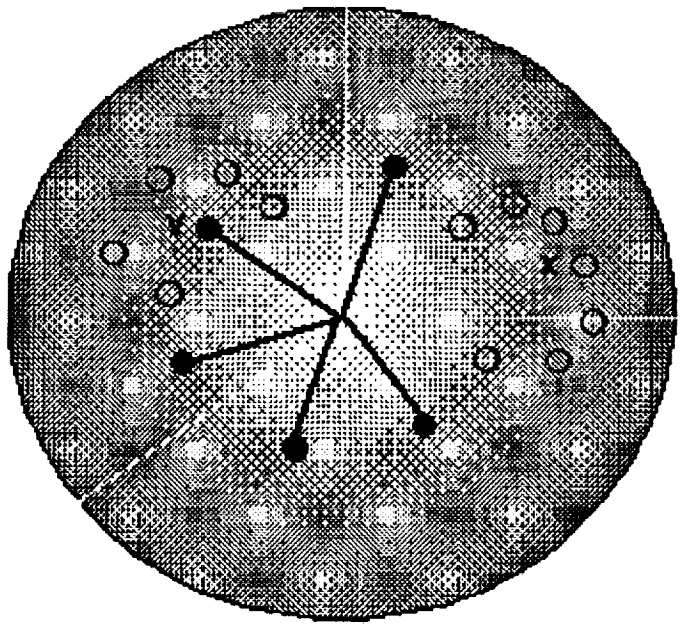
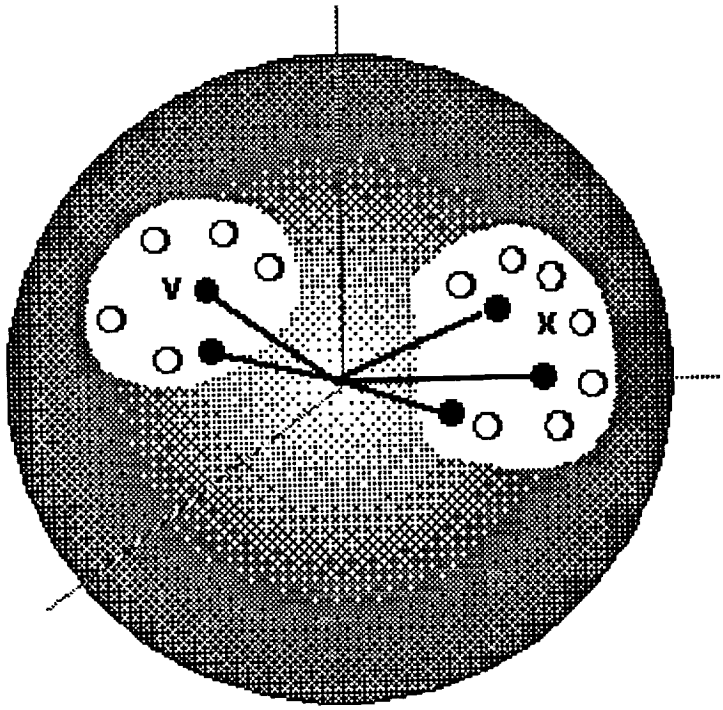


Figure 9. Terminal Weight Vectors and an HCM Partition in the KSO Scheme



4. DISCUSSION AND CONCLUSIONS

First, we itemize the major *differences* between ISODATA and KSO :

(D1) FCM , HCM and SHCM are *intrinsic* clustering methods - i.e., one of their inputs is an unknown partition, and one of their outputs is a partition of unlabeled data set X which is optimal in the sense of minimizing a norm driven objective function. The KSO method, on the other hand, needs an *a posteriori* rule such as the nearest prototype rule at (9) to generate a partition of the data non-iteratively. We might call this an *extrinsic* clustering scheme. Moreover, without labeled data that can be used to discover which subsets of the c^* multiple prototypes found by the KSO scheme should be identified with each of the c classes assumed in (9), there is no general way to even implement (9) with the KSO rule. Thus, much must be added to KSO to make it a true clustering method.

(D2) The data set X is used differently. KSO uses the data sequentially (locally) and hence, its outputs are dependent on local geometry and order of labels, whereas ISODATA utilizes the data globally, and

updates both the weights and partition values in parallel at each pass. In this sense KSO is most akin to Sequential Hard c-Means, which is also sensitive to ordering of labels - this is often regarded as a fatal flaw in clustering.

(D3) KSO can have multiple prototypes for each class; ISODATA has but one. In clustering, the usual assumption is that c is unknown, and one resorts to various cluster validity schemes to validate the results of any algorithm. Since the KSO scheme uses many prototypes, without assuming an underlying "true but unknown" number of clusters, this is advantageous to the user. However, the dilemma of how to convert the prototypes into clusters, as discussed in (D1), persists.

(D4) KSO uses local orientation ($\cos\theta = \langle \mathbf{x},\mathbf{v} \rangle$) on the unit ball as the measure of similarity between data and weights, whereas ISODATA uses cluster shape (via the eigenstructure of A) and location (via the lengths of the weights and the data) to assess (dis)similarity between the data and prototypes. Thus, the c-Means approach has a much more "statistical" flavor than KSO. On the other hand, KSO uses the dot product at each node, in the spirit of the McCulloch-Pitts neuron. Thus, local computations in the KSO scheme proceed on the basis assumed by many workers in neural network research, and make the KSO scheme more easily identifiable with this type of computational architecture.

(D5) KSO preserves "order" in a certain sense; ISODATA does not. This property of the KSO method is perhaps its most interesting distinction. There is little hope that c-Means has a similar property. Since cognitive science assures us that one aspect of intelligence is its inherent ability to order, this aspect of the KSO approach again shows well in its favor. A significant line of research concerns whether or not the FCM/HCM models possess this, or any similar property.

(D6) Weight updates in the KSO method are intuitively appealing; weight updates in ISODATA are mathematically necessary. Since the update formula in c-Means finds either real or generalized centroids, we might claim that this scheme is also intuitively appealing. In this regard the c-Means algorithms (including SHCM) have a clear theoretical advantage, at least in terms of justification of the procedure used.

(D7) FCM, HCM and SHCM are all well-defined optimization problems; KSO is an heuristic procedure. An interesting question about KSO is this: what function is being optimized during iteration? An answer to this question would be both useful and illuminating. The criterion functions that drive FCM, HCM and SHCM are well understood geometrically and statistically; discovery of a criterion function for Kohonen's algorithm might supply a great deal of insight about other properties of the algorithm and its outputs.

(D8) KSO partitions have so far been generated with the nearest prototype rule and the Euclidean norm, whereas FCM, HCM and SHCM can be used with any inner product and two Minkowski norms. Much research can be done on the issue of how best to use the Kohonen prototypes to find cluster substructure. There are many natural ways besides the nearest prototype rule to use KSO outputs with the weights $\{\mathbf{v}_i\}$. For example, one could simply distribute unit memberships satisfying (1b) across the KSO nodes at each step using distance proportions. This generalizes Kohonen's model from a "neighborhood take all" to a "neighborhood share all" concept. One certainly suspects that it is possible to incorporate $U \in M_{fcn}$ as an *unknown* in the KSO approach , so that an extended KSO algorithm creates partitions of the data that are necessary, rather than, as in the current use of the HCM labeling rule, a heuristic afterthought.

Major *similarities* between ISODATA and KSO include:

(S1) If we let $(U_F, \mathbf{v}_F)$, $(U_H, \mathbf{v}_H)$, $(U_S, \mathbf{v}_S)$, and $(U_K, \mathbf{v}_K)$ denote, respectively, the pairs found by FCM, HCM, SHCM and KSO, we note that $(U_F, \mathbf{v}_F)$ is a critical point for J_m, while $(U_H, \mathbf{v}_H)$, $(U_S, \mathbf{v}_S)$, and $(U_K, \mathbf{v}_K)$ are, because of the HCM theorem, (possibly different) critical points of J_1. However, $(U_H, \mathbf{v}_H) \neq (U_S, \mathbf{v}_S) \neq (U_K, \mathbf{v}_K)$ generally. This suggests that (i) HCM (and especially SHCM) and KSO as described herein are most definitely related, and (ii), there should be a generalized (fuzzy) KSO that bears the same relationship to FCM that the hard c-Means versions bear to the current version of KSO. It seems clear that there is a stronger mathematical link between FCM/HCM and KSO than is currently known. Connection of the two approaches begins with careful formulation of a constrained optimization problem that holds for KSO. This involves finding a global KSO criterion function and necessary conditions that *require* the calculation of the weight vectors $\{\mathbf{v}_i\}$ as in KSO <3b>.

(S2) Both algorithms find prototypes (weights or cluster centers) in the data that provide a compressed representation of it, and enable nearest prototype classifier design. Recent work by Huntsberger and Ajjimarangsee [13] indicates that FCM is at least as good as KSO in terms of minimizing apparent error rates. And further, FCM sometimes generates identical solutions to KSO on various well known data sets. This is another powerful indicator of the underlying (unknown) relationship between the KSO and c-Means methods. Much can be done empirically to confirm or deny specific relationships between the two methods.

We have itemized some similarities and differences between two approaches to the clustering of unlabeled data - Hard/Fuzzy c-Means and Kohonen's self-organizing feature maps (KSO), and posed some questions concerning each method. Successful resolution of these questions will benefit both models. Numerical convergence properties and the neural-like behavior of both the extended KSO and FCM algorithms should

be established. Issues to be studied should include : robustness, adaptivity , parallelism , apparent error rates, time and space complexity, type and rate of convergence, optimality tests, and initialization sensitivity.

5. REFERENCES

[1] Kohonen, T. Self-Organization and Associative Memory, 3rd Edition, Springer-Verlag, Berlin, 1989.

[2] Bezdek, J. Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York, 1981.

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

[4] Pao, Y.H. Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading, 1989.

[5] Lippman, R. An Introduction to Neural Computing, *IEEE ASSP Magazine*, April, 1987, 4-22.

[6] Ball, G. and Hall, D. A Technique for Summarizing Multivariate Data, Behav. Sci., 12, 1967, 153-155.

[7] Dunn, J.C. A Fuzzy Relative of the ISODATA Process, *Jo. Cybernetics*, 3, 1974, 32-57.

[8] Gustafson, D. and Kessel, W. Fuzzy Clustering with a Fuzzy Covariance Matrix, in Proc. IEEE CDC, 1978, 761-766.

[9] Bezdek, J. C., Coray, C., Gunderson, R. and Watson, J. Detection and Characterization of Cluster Substructure, I and II, *SIAM Jo. of Appl. Math.*, 40(2), 1981, 339-372.

[10] Pedrycz, W. Algorithms of Fuzzy Clustering with Partial Supervision, *Patt. Recog. Letters*, 3, 1985, 13-20.

[11] Dave, R. Fuzzy Shell Clustering and Applications to Circle Detection in Digital Images, in press, *Int'l. Jo. of General Systems*, 1989.

[12] Bobrowski, L. and Bezdek, J. c-Means Clustering with the ℓ_1 and ℓ_∞ Norms, in review, IEEE Trans. SMC, 1990.

[13] Huntsberger, T. and Ajjimarangsee, P. Parallel Self-Organizing Feature Maps for Unsupervised Pattern Recognition, in press, *Int'l. Jo. General Systems*, 1990.