MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1390 January, 1993

C.B.C.L. Paper No. 079

Some Extensions of the K-Means Algorithm for Image Segmentation and Pattern Classification

Jose L. Marroquin and Federico Girosi

Abstract

In this paper we present some extensions to the k-means algorithm for vector quantization that permit its efficient use in image segmentation and pattern classification tasks. It is shown that by introducing state variables that correspond to certain statistics of the dynamic behavior of the algorithm, it is possible to find the representative centers of the lower dimensional manifolds that define the boundaries between classes, for clouds of multi-dimensional, multi-class data; this permits one, for example, to find class boundaries directly from sparse data (e.g., in image segmentation tasks) or to efficiently place centers for pattern classification (e.g., with local Gaussian classifiers). The same state variables can be used to define algorithms for determining adaptively the optimal number of centers for clouds of data with space-varying density. Some examples of the application of these extensions are also given.

Copyright © Massachusetts Institute of Technology, 1993

This report describes research done within CIMAT (Guanajuato, Mexico), the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041; and by a grant from the National Institutes of Health under contract NIH 2-S07-RR07047. Additional support is provided by the North Atlantic Treaty Organization, ATR Audio and Visual Perception Research Laboratories, Mitsubishi Electric Corporation, Sumitomo Metal Industries, and Siemens AG. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. J.L. Marroquin was supported in part by a grant from the Consejo Nacional de Ciencia y Tecnologia, Mexico.

1 Introduction

Finding a set of representative vectors for clouds of multi-dimensional data is an important issue in data compression [9], signal coding [9, 8], pattern classification [5] and function approximation tasks [22, 24]. These centers are said to be *representative* in the following sense: given a set of N points in D–dimensional Euclidean space, $X = \{x_i, i = 1, \ldots N\}$, a set of M centers $\{m_1, \ldots, m_M\}$, $m_k \in \mathbf{R}^D, k = 1, \ldots, M$, partitions X into M sets $\{S_1, \ldots, S_M\}$, where each set S_k corresponds to those points in X inside the Voronoi polytope [8] of center m_k:

$$S_k = \{x_i : \|x_i - m_k\| < \|x_i - m_j\|, j \neq k\}$$

where $\|\cdot\|$ is the Euclidean norm. The set of centers is *optimal* with respect to this norm, if they minimize the error measure:

$$\begin{aligned} \mathcal{E}(m) &= \frac{1}{2}\sum_{k=1}^{M}\sum_{x_i \in S_k} \|x_i - m_k\|^2 = \qquad (1) \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{M} \|x_i - m_k\|^2 \mathbf{1}_{S_k}(x_i) \end{aligned}$$

where $\mathbf{1}_S(\cdot)$ is the indicator function of set S.

A widely used technique for finding a locally optimal set of M centers is the *k-means algorithm* [18]: one starts with a random center configuration $m^{(0)}$ which is then updated using the rule:

$$m_k^{(t+1)} = \frac{\sum_{i=1}^{N} x_i \mathbf{1}_{S_k^{(t)}}(x_i)}{\sum_{i=1}^{N} \mathbf{1}_{S_k^{(t)}}(x_i)} \qquad (2)$$

where

$$S_k^{(t)} = \{x_i : \|x_i - m_k^{(t)}\| < \|x_i - m_j^{(t)}\|, j \neq k\}$$

It has been shown [1, 17] that this algorithm in fact converges to a local minimum of (2). For our purposes, however, it is more convenient to use a different form of this algorithm, in which each data point is used in turn to update the corresponding center location [15]. If at time t point x_i is selected, we put:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t(x_i - m_k^{(t)}), \text{ if } i \in S_k^{(t)} \qquad (3) \\ &= m_k^{(t)}, \text{ otherwise} \end{aligned}$$

where $\{a_t\}$ is a non–increasing sequence of scalars. This scheme, which we will call the Local K-means Algorithm (LKMA) has the obvious advantage of being able to operate in a dynamic environment, where data are continuously arriving. It also allows for generalizations that produce a "self–organizing" behavior of the centers (see below). However, if the spatial distribution of the data is non–uniform (i.e., if the data are clustered) its performance may be poor. In section 2 we present an extended version of this scheme which is more amenable to formal analysis.

A straightforward generalization of this technique may be also used for classification purposes [15]: if one is given for each data point x_i an associated class $C(x_i) \in \{c_1 \ldots, c_L\}$, one may find M_j optimal centers for each class j : $\{m_{jk}, j = 1, \ldots, L, k = 1, \ldots, M_j\}$ using:

$$\begin{aligned} m_{jk}^{(t+1)} &= m_{jk}^{(t)} + a_t(x_i - m_{jk}^{(t)}), \qquad (4) \\ &\quad \text{if } x_i \in S_{jk}^{(t)} \text{ and } C(x_i) = j \\ &= m_k^{(t)}, \text{ otherwise} \end{aligned}$$

where $S_{jk}^{(t)}$ is the set of data points inside the Voronoi polytope of $m_{jk}^{(t)}$. This procedure is similar to the LVQ1 classification method [15] (except that in that case, one introduces a repulsion term from those points with $C(x_i) \neq j$ that are inside S_{jk} to push the centers away from the decision boundaries) and is equivalent to the use of update rule (3) for each class in a decoupled fashion. After convergence, a specimen y will be assigned to class j if for some k ,

$$\|y - m_{jk}\| < \|y - m_{ln}\| \text{ for all } l, n$$

Although this technique will work in general, it has a relatively low efficiency, since most of the centers are irrelevant for the classification task: only those that are close to the inter–class boundaries will play an effective role. In fact, we will show how to build more efficient classifiers by finding centers that are representative of the set of *class boundaries* instead of the set of class interior points. This will be discussed in the following section.

Another modification of the basic procedure (3) allows one to give a neighborhood structure to the set of centers. Thus, Kohonen's *Self Organizing Maps* [15] show how a 1 or 2–dimensional lattice structure may be imposed to the set of centers, and how this structure may, in many cases, reflect the internal organization of the data. If the neighborhoods of each center m_k at time t are defined as sets of centers $N_k(t)$ that satisfy:

$$m_j^{(t)} \in N_k^{(t)} \Leftrightarrow m_k^{(t)} \in N_j(t)$$

$$m_k^{(t)} \notin N_k(t)$$

we get the following modified update equations:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t(x_i - m_k^{(t)}), \text{ if } x_i \in S_k^{(t)} \qquad (5) \\ &= m_k^{(t)} + a_t(x_i - m_k^{(t)})h(\|m_k^{(t)} - m_j^{(t)}\|), \\ &\quad \text{if } x_i \in S_j^{(t)}, j \neq k, \text{ and } m_k \in N_j(t) \\ &= m_k^{(t)}, \text{ otherwise} \end{aligned}$$

where $h(\cdot)$ is a decreasing function. In Kohonen's work, the neighborhoods $\{N_k(t)\}$ are initially very large and shrink slowly to their final desired size (e.g., a nearest neighbor structure).

This scheme suffers from some limitations: in the first place, it is difficult to analyze (except in some particular cases [25]), and thus to understand its performance in a precise way; besides, the neighborhood structure is *imposed* rather than found from the data (although some modifications have been proposed to this end [21]), which limits its usefulness in unsupervised clustering tasks.

The above considerations provide the motivation for the present work: it is our purpose to extend the LKMA so that some of its limitations are overcome; specifically, we will propose extended versions of the algorithm that:

i) Allow for a rigorous analysis of its convergence properties.

ii) Work well for clustered data.

iii) Will, if desired, find the centers of the inter–class boundary set.

iv) Adapt the number of centers to the local spatial density of the data.

v) Find the "natural" neighborhood structure for the centers of a data set (i.e., may be used for unsupervised clustering).

The plan of the presentation is as follows: in section 2, we introduce a family of algorithms that include the LKMA as a limiting case, and give a general convergence theorem for this class; also in this section, we present some basic extensions and generalizations needed for finding the centers of the inter–class boundary set, and for adapting the number of centers to the data density. In section 3 we give examples of the application of the extended scheme, specifically, to image segmentation and pattern classification, and finally, in section 4, we present some conclusions and open problems.

2 Extended Local K-Means Algorithms

First, we will introduce a generalization of the LKMA that will help us to understand its convergence properties. Consider again a set $X = \{x_1, \ldots, x_N\}$ of points in $\mathbf{R}^D$, a set of centers $\{m_1, \ldots, m_M\}$, and the weighted error measure:

$$\mathcal{E}_\beta(m) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{M} \|x_i - m_k\|^2 w_{ik}^\beta \tag{6}$$

where

$$w_{ik}^\beta = \frac{\exp[-\beta\|x_i - m_k\|^2]}{\sum_{j=1}^{M}\exp[-\beta\|x_i - m_j\|^2]} \tag{7}$$

It is clear that the error measure (2) may be obtained as the limit of (6) as $\beta \to \infty$. Now, $\mathcal{E}_\beta$ is differentiable, and its gradient with respect to m_k is given by:

$$\nabla_k \mathcal{E}_\beta = \sum_{i=1}^{N}(m_k - x_i)W_{ik}^\beta \tag{8}$$

where

$$W_{ik}^\beta = w_{ik}^\beta(1 + \beta(\sum_j w_{ij}^\beta\|x_i - m_j\|^2 - \|x_i - m_k\|^2))$$

A standard gradient descent procedure for minimizing $\mathcal{E}_\beta$ would therefore take the form:

$$m_k^{(t+1)} = m_k^{(t)} - a_t\sum_{i=1}^{N}(m_k^{(t)} - x_i)W_{ik}^\beta \ . \tag{9}$$

for some sequence $\{a_t\}$ converging to zero. However, it is often more convenient to adopt a *stochastic gradient descent* algorithm for minimizing $\mathcal{E}_\beta$, that is equivalent to approximating, at each step, the sum on the right side of eq. (9) with just one term, randomly drawn among the N terms. In formula:

$$m_k^{(t+1)} = m_k^{(t)} - a_t(m_k^{(t)} - x_{\xi_t})W_{\xi_t k}^\beta \tag{10}$$

where $\{\xi_t\}$ is a sequence of random variables which take values on $\{1, 2, \ldots, N\}$. This minimization technique is especially convenient when the data points x_i come one at a time, and it has been extensively used by the neural network community, as a part of the so called "back-propagation" procedure for neural networks training. In the limit as $\beta \to \infty$, (10) becomes precisely the LKMA (3). The convergence of (10) to a local minimum of $\mathcal{E}_\beta$ follows from the following lemma (the proof is given in the appendix; see also [28] for closely related results):

Lemma 2.1 *Let $F(y) : \mathbf{R}^k \mapsto \mathbf{R}$ be of the form:*

$$F(y) = f_0(y) + \frac{1}{N}\sum_{i=1}^{N} f_i(y)$$

where f_i are differentiable functions whose gradient is bounded and satisfies the following Lipschiz condition:

$$\|\nabla f_i(y) - \nabla f_i(y')\| \leq M\|y - y'\| \ \ i = 0, \ldots, N$$

for some positive number M. Let $\{y_n\}$ be the sequence

$$y_{n+1} = y_n - a_n(\nabla f_0(y_n) + \nabla f_{\xi_n}(y_n)) + a_n\eta_n \tag{11}$$

where $\{\xi_n\}$ is a sequence of random variables that take values in $\{1, \ldots, N\}$ with uniform probability distribution, $\{a_n\}$ is a sequence of scalars satisfying:

$$\sum_{i=1}^{\infty} a_n = \infty \quad \text{and} \quad \sum_{i=1}^{\infty} a_n^2 < \infty$$

and $\{\eta_n\}$ is a sequence of random variables that is bounded and converges to zero with probability one. Assume that $\{y_n\}$ is bounded and that S is a locally asymptotically stable point of the ordinary differential equation:

$$\dot{y} = -\nabla F \tag{12}$$

with domain of attraction A_S. Then, if $y_n \in G$ for all n, for some compact domain $G \subset A_S$, $\{y_n\}$ converges to S with probability one.

To satisfy the conditions of the lemma, the sequence $\{a_t\}$ in (10) must be chosen appropriately. One possible choice is, for example,

$$a_t = 1/t \ .$$

2.1 Soft Winner–Takes–All K–Means

Algorithm (10) may be understood as a "soft" version of the WTA scheme that implements (2): if β is relatively small, when a new data point arrives, it will update the position of not only the closest center, but of others that are close by as well.

A similar result may be obtained by using, instead of (6), an information–related distorsion measure [2]:

$$\mathcal{E}_\beta(m) = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\sum_{k=1}^{M}\exp[-\beta\|x_i - m_k\|^2]\right] \quad (13)$$

This error measure also corresponds to the log likelihood function of a Gaussian mixture model for the data distribution [10] [23], where: the means of the Gaussians correspond to the center locations; the proportions are all equal to $1/N$, and all the covariance matrices are equal to $\beta^{-1}I$.

In this case, the corresponding stochastic gradient descent equation takes the form:

$$m_k^{(t+1)} = m_k^{(t)} - a_t(m_k^{(t)} - x_{\xi_t})w^{\beta}_{\xi_t k} \quad (14)$$

where w^{β}_{ik} is given by (7). In the limit as $\beta \to \infty$ both (10) and (14) give the same WTA update equation; their experimental behavior is also very similar.

These schemes are also related to the self–organizing maps (5), except that in this case the neighborhood of each center is not predetermined, but rather, it varies dynamically as the iterations proceed: the relative values of the weights $\{W^{\beta}_{ik}\}$ for each i are related to the instantaneous neighborhood structure of the centers; we may define the instantaneous link status of centers j and k as:

$$l^{(i)}_{jk} = \sum_{i=1}^{N} W^{\beta}_{ij}W^{\beta}_{ik}$$

and the "natural" neighborhood structure for a given data set by the average $\bar{l}_{jk}$ of $l^{(i)}_{jk}$ over all i: two centers (j,k) may be considered neighbors if $\bar{l}_{jk} > \theta$, for some appropriate threshold θ (a similar, although computationally more expensive scheme may be found in [21]). Figure 1 shows this natural neighborhood structure for several two–dimensional data sets. As one can see, it represents adequately the inner structure of the data, and so, it may be used for unsupervised clustering tasks.

———— Figure 1 around here ————

This soft–WTA algorithm may also be useful for improving the performance of the LKMA with clustered data. To this end, one may use a time–varying β, rather than a constant one: starting with a relatively small value ensures that every center will be attracted to some data cluster, regardless of its (random) initial position; the value of β may then be increased to obtain a final configuration that minimizes the non–weighted cost function (2). One has to be careful, however, when working with small values of β, so that the center positions are kept different from each other, since if two centers collapse, they will remain in that state regardless of an increased β (if they are updated in parallel). This may be accomplished by adding a β–dependent random component to the update equation:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t(x_i - m_k^{(t)}) + \frac{\epsilon}{\beta}r(t), \quad (15)\\ &\quad \text{if } i \in S_k^{(t)} \\ &= m_k^{(t)}, \text{ otherwise} \end{aligned}$$

where $r(t)$ is a uniformly distributed unit D–vector and ϵ is a small number (the convergence of this modified scheme also follows from lemma 1). The relative performance of this scheme, compared with the standard LKMA is illustrated in figure 2.

———— Figure 2 around here ————

2.2 State–Augmented LKMA

Another way of improving the performance of the standard LKMA, is to include in the state of each processor (center) statistics about its past dynamic behavior. Specifically, one may keep track of the number of times it has "won" over the other processors:

$$h_k(t) = \sum_{\tau=1}^{t} \mathbf{1}_{S_k^{(\tau)}}(x_{i(\tau)}) \quad (16)$$

where $i(t)$ is the index of the data point selected at time t, and (m_k, h_k) is the augmented state of processor k.

For multi–class data, one may use specific centers for each class, and include, for each one of them, another state variable $\hat{h}$ that counts the number of times the class–specific processor has "won" with a data point belonging to its class. Thus, if m_{jk} is the k^{th} center for class j, one has:

$$\hat{h}_{jk}(t) = \sum_{\tau=1}^{t} \mathbf{1}_{S_{jk}^{(\tau)}}(x_{i(\tau)}) \cdot \delta(C(x_{i(\tau)}) - j) \quad (17)$$

where $C(x_i)$ is the class to which x_i belongs and $\delta(\cdot)$ is the standard Kroneker delta function.

We will now indicate how to modify the update rule, so that this information is taken into account.

2.2.1 Adaptive Number of Centers

This augmented state information may be used for adapting the number of centers to the local density of the data points. This may be accomplished by updating the center configuration after each full sweep over the data set (or after a sufficiently large number of data has come in), supressing those centers that have won very few times, and splitting those that have won too many times. Specifically:

For each center k:

$$\text{if } h_k < \theta_i N, \textit{ supress the center.} \quad (18)$$

else if $h_k > \theta_s N$, generate a new center at a location corresponding to one the data points inside the current Voronoi polytope of center k.

where $\theta_i, \theta_s \in [0,1]$ are two suitably chosen thresholds such that $(\theta_s - \theta_i)N > 1$. Note that this choice for the location of the new centers is necessary to ensure convergence (see below); in practice, however, one may simply locate them at $m_k + \epsilon r$ where r is a random unit vector, and ϵ a positive number small enough, so that the new center is attracted by at least one data point inside S_k.

If the total number of centers change after a sweep, one should reset h_k to zero for all k, and a_t to a_{t-N}, and effect a new sweep until the number of centers stabilize.

It is not difficult to show the convergence of (18) when the lower threshold θ_i is set to 0: in this case, one starts with one processor (center) and successively generate new ones until the Voronoi polytopes corresponding to all centers contain less than $\theta_s N$ data points. Since in this case the number of centers increases monotonically and it is bounded above, (e.g., by the total number of data points), it will necessarily converge to a fixed number M^*.

θ_s is a free parameter that controls the expected average number of points per center, while θ_i controls the variance of this number (a small variance is obtained if $(\theta_s - \theta_i)$ is small). The fact that one has control over the variance means that one can generate more uniform center distributions with this method than with the standard k-means scheme. This, in turn, will usually improve significantly the performance of other procedures that may use these centers, for example, for vector quantization or for function approximation (see section 3).

In practice, it is convenient to set the lower threshold to a positive value to prevent centers to be attracted to single outlier data points, as well as the existence of centers with empty Voronoi polytopes (which may happen due to random initialization, if one starts with more than one center). Note, however that convergence cannot be guaranteed in this case; consider the following example: suppose we have $N = 7$ data points; $N\theta_s = 5$ and $N\theta_i = 4$. It is clear that the number of centers generated by (18) will always oscillate between 1 and 2.

This kind of pathological situations are unlikely to occur in practice though, specially if $(\theta_s - \theta_i)$ is large enough (say, if $\theta_s > 3\theta_i$). A practical way of ensuring convergence in any case is to let θ_i go to zero after a fixed number of iterations.

2.2.2 Boundary Finders

In the case of multi–class data, the augmented state may be used to find the inter–class boundaries directly from sparse data.

Let us assume that we have class–specific centers for classes 1 through $M-1$. Consider the 2–class case first: the augmented state will contain, for each center k, the vector $(m_k, h_k, \hat{h}_{1k})$ (we only have one type of centers in this case). The idea is to constrain the update rule so that a center position is updated approximately the same number of times by data points belonging to each class, so that at all t,

$$\hat{h}_{1k}^{(t)} \approx \hat{h}_{2k}^{(t)} \approx \frac{1}{2} h_k^{(t)}$$

The boundary–finding update rule may thus take the form:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t(x_i - m_k^{(t)}), \\ &\quad \text{if } x_i \in S_k^{(t)} \text{ and } \hat{h}_{1k}^{(t)} \le \frac{1}{2} h_k^{(t)} \\ &= m_k^{(t)} \text{ , otherwise} \end{aligned} \tag{19}$$

where x_i is the data point chosen at time t.

It is clear that with this rule we will have, at any time t, $\mid h_k^{(t)} - 2\hat{h}_{1k}^{(t)} \mid \le 1$, so that there will be approximately the same number of data points belonging to each class inside the Voronoi polytope of each center, provided that the data density is uniform. In this case, upon convergence, every center k will be located at about the mid-point of the centroids of the sets $\{\mathcal{C}_0 \cap S_k\}$ and $\{\mathcal{C}_1 \cap S_k\}$ where $\mathcal{C}_n$ is the set of data points of class n.

To see why this is true, note that when the update rule (20) reaches its steady state, we must have that

$$E[(x_i - m_k)] = \sum_{c=1}^{2} \sum_{x_i \in S_k} (x_i - m_k) P_k(i,c) = 0$$

where $P_k(i,c)$ is the probability of selecting an example i that is in S_k and belongs to class c and $E[\cdot]$ denotes the expected value. Now,

$$\begin{aligned} P_k(i,c) &= \Pr(\text{select } i \mid i \in \mathcal{C}_c \cap S_k) \Pr(\text{select } \mathcal{C}_c) \approx \\ &\approx \frac{1}{\mid \mathcal{C}_c \cap S_k \mid} \cdot \frac{1}{2} \end{aligned} \tag{20}$$

where $\mid \mathcal{C}_c \cap S_k \mid$ denotes the number of points of class c inside S_k, so that

$$m_k \approx \frac{1}{2} \sum_{c=1}^{2} \frac{1}{\mid \mathcal{C}_c \cap S_k \mid} \sum_{i \in \mathcal{C}_c \cap S_k} x_i$$

It is in this sense that one may say that the centers are representative samples of the inter–class boundary set.

This procedure may be generalized to $Q > 2$ classes, by sampling, for each class $\{1, \ldots, Q-1\}$, the boundary between itself and all the other classes. This however, is not very efficient, since many parts of the boundary will be sampled several times. A more economical sampling may be obtained by defining the sets:

$$T_0^k = \mathcal{C}_k$$

$$T_1^k = \mathcal{C}_{k+1} \cup \mathcal{C}_{k+2} \cup \ldots \mathcal{C}_Q$$

for $k = 1, \ldots Q-1$, and finding, for each k, the centers $\{m_{k1}, \ldots, m_{kM_k}\}$ that sample the boundary between the sets T_0^k and T_1^k using algorithm (20).

It is of course possible (and desirable) to combine this procedure with the one for finding the number of centers in an adaptive way. Figure 3 shows the performance of this combined scheme for binary–class data in 2 dimensions. Other examples of applications of this algorithm are presented in section 3.

_______________ Figure 3 around here _______________

2.3 Arbitrary Neighborhood Structures and Smoothness Constraints

We presented in section 1 (equation (5)) Kohonen's modification to the LKMA, which produces "self–organizing maps" between the data points and the centers, that is, center configurations with a prescribed neighborhood structure. These maps are interesting because they provide a model for the topology–preserving mappings that are known to exist between sensory inputs and the brain cortex [6, 14].

Kohonen's algorithm, however, presents a number of problems: firstly, it is difficult to analyze (its convergence and the properties of its fixed points have not been established in the general case); secondly, its experimental rate of convergence is usually very slow, and finally, it is not clear how to extend it to include other desired properties of the center configurations (i.e., the requirement that the centers lie in a smooth curve, etc.).

In this section we will present an alternative algorithm for producing organized center structures which can be derived from a Bayesian formulation of the problem, with a Gibbsian prior for the center configurations. This approach not only permits a cleaner analysis of the algorithm, but also exhibits faster convergence behavior, and can be easily generalized to include other properties (e.g., smoothness) of the configurations.

The basic idea in this approach is to express the prior constraint on the organization of the centers in probabilistic terms, specifically, in the form of a discrete Markov Random Field (MRF) model [20, 13, 3, 26, 7], in which the center locations correspond to the state variables associated with the nodes of a graph whose topology is related —but not necessarily identical— to the desired neighborhood structure of the centers; in fact this structure, as well as the smoothness of the locations of the center configurations will depend both on the graph topology and on the particular choice of the potential functions that define the model.

The prior probability distribution of the center configuration will thus be of the form:

$$\pi(m) = \frac{1}{Z} e^{-U(m)}$$

where Z is a normalizing constant and the energy U is of the form:

$$U(m) = \sum_C V_C(m)$$

where the sum is taken over the cliques $\{C\}$ of the corresponding neighborhood system and $V_C(\cdot)$ are *potential functions* that depend only on the values of the state variables associated with the nodes contained in each clique.

The optimal center configuration may now be defined as the most likely one (given the prior MRF structure) that minimizes the error criterion (6), i.e., as the Maximum a Posteriori (MAP) estimator of a Gibbsian field with posterior energy given by:

$$U_P(m) = \frac{1}{2} \sum_i \sum_k \|x_i - m_k\|^2 w_{ik}^\beta + \lambda \sum_C V_C(m) \quad (21)$$

where λ is a positive, real parameter.

If the potential functions are differentiable, the following update rule will converge to a local minimum of (21):

$$m_k^{(t+1)} = m_k^{(t)} + a_t \left[(x_i - m_k^{(t)}) W_{ik}^\beta - \lambda \sum_{C:k\in C} \frac{\partial V_C(m^{(t)})}{m_k} \right] \quad (22)$$

for an appropriate choice of the sequence $\{a_t\}$ (see lemma 1).

Note that in the particular case of cliques of size 2, and quadratic potentials of the form:

$$V_{jk}(m) = \|m_j - m_k\|^2$$

the posterior energy corresponds to the composite cost function discussed in [2], and the corresponding update equation, for first order (nearest–neighbor) systems, reduces to the algorithm proposed by Durbin and Mitchison [6] for the development of cortical maps.

Self–organization of the centers in rectangular lattices may be obtained using second order neighborhoods and cliques of size 3 that correspond to triads of neighboring sites that lie in the same row (or column) of the lattice, so that 3 centers belonging to the same clique will not contribute to the energy if they lie in a straight line.

If the lattice size is relatively large, however (greater than about 8×8), the system (22) with this type of potentials will very often converge to local minima for which the lattice appears "folded" in some way, so that the global order is not properly established (see figure 4). This can be remedied in two ways: first, it is necessary to include potentials that assign high energies to folded configurations; a simple choice is to assign to cliques of 4 sites $\{i, j, k, l\}$, that lie in the corners of unit squares in the lattice, potentials of the form:

$$\frac{1}{4}[\|m_i - m_j\|^2 - \|m_k - m_l\|^2]^2$$

where $(i, j), (j, k), (k, l)$ and (l, i) are nearest neighbors. Secondly, it is necessary to define a neighborhood system that spans a wide range of scales, with potentials that increase their relative weights with the scale size. Thus, for example, if $m_{i,j}$ is the center at row i and column j of the lattice, the multi–scale potentials that correspond to column alignment may take the form:

$$V_{ij}^{(k)}(m) = \lambda_k \| - m_{i,j-k} + 2m_{i,j} - m_{i,j+k}\|^2 \quad (23)$$

with $\lambda_{k+1} > \lambda_k$, for $k = 1, \ldots, K$.

Figure 4 around here

A more efficient way of achieving the same result is to define a "pyramid" of processes that operate from coarse to fine scales, and that increase the number of centers at each refining step: one may start with a 3×3 lattice , which after a few iterations of (22) may be refined by adding intermediate centers whose initial positions correspond to the midpoints of existing ones (see figure 5), and repeat the whole procedure recursively until the desired number of centers is obtained. Note that with this procedure, the potentials are always of the form (23) with $k = 1$.

Figure 5 around here

The final configurations obtained in this way (see figure 4–b) are very similar to those obtained with Kohonen's algorithm [15] (which also incorporates long range interactions), but since the neighborhood size remains fixed, the computational complexity is lower, and since the new centers are already close to their correct (globally ordered) positions, the convergence rate is significantly faster.

It is convenient, as in the case of Kohonen's scheme, to mantain a fixed, relatively large a_t in (22) until the final number of centers is reached. At this point, the use of an appropriately decreasing sequence guarantees the final convergence to a local minimum of (21) (see lemma 1). Other examples of the use of this approach will be given in the next section.

3 Applications

In this section we present some applications that illustrate the power of the techniques we have developed

3.1 Edge Detection from Sparse Data

Suppose we have a set of sparse data points $X = \{x_1, \ldots, x_N\}$ inside the unit square Ω in $\mathbf{R}^2$, which may belong to either one of two classes $\{0, 1\}$, and suppose there is a closed region $A \subset \Omega$ whose boundary is a closed, smooth curve, and such that

$$C(x) = 1 \Leftrightarrow x \in A$$

(i.e., all the data points in class 1 are inside A). The problem now is to find a polygonal line (i.e., a *sequence* of points $\{m_1, \ldots m_M\}$) that lies close to the smooth curve that defines the boundary of A.

This may be achieved by combining the adaptive boundary–finding scheme of section 2.2.2 with a prior MRF constraint on the configuration of centers that corresponds to a circular lattice (i.e., a closed polygonal line). In particular, to every clique of 3 neighboring sites (i, j, k) we associate the potential:

$$V_{ijk}(m) = \frac{1}{2}\| - m_i + 2m_j - m_k\|^2$$

(note that m_1 and m_M are considered neighbors in a circular lattice).

The combined update rule takes the form:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t \left[(x_i - m_k^{(t)}) - \lambda \sum_{C:k\in C} \frac{\partial V_C(m^{(t)})}{\partial m_k} \right], \\ &\quad \text{if } x_i \in S_k^{(t)} \text{ and } \hat{h}_k^{(t)} \le \frac{1}{2} h_k^{(t)} \\ &= m_k^{(t)} - a_t \left[\lambda \sum_{C:k\in C} \frac{\partial V_C(m^{(t)})}{\partial m_k} \right], \\ &\quad \text{if } x_i \in S_j^{(t)} \text{ and } \hat{h}_j^{(t)} \le \frac{1}{2} h_j^{(t)}, j \ne k \\ &= m_k^{(t)} \text{ , otherwise} \end{aligned} \quad (24)$$

where x_i is the data point chosen at time t.

An example of the performance of this algorithm is shown in figure 6. Note that since in the final configuration the centers are ordered, this scheme is in fact finding the (discrete) boundary curve from sparse data without interpolating the corresponding surface. In this sense, it may be said that this algorithm finds the initial position, the number of knots and the final configuration of a "snake" [12] that approximates the inter–class boundary.

Figure 6 around here

Figure 7 around here

With straightforward modifications, this scheme may be used for finding: multiple closed boundaries (fig. 7–a and 7–c); open curves that go from one border of the image to another (fig. 7–b and 7–d), etc.

3.2 Local Gaussian Classifiers

Gaussian Classifiers [5] are a well known class of procedures for the segmentation of multi–class, multi–dimensional data. The classification procedure for each specimen $x \in \mathbf{R}^n$ involves the computation of Q quadratic discriminant functions (one for each class) :

$$\begin{aligned} D_k &= (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \mid \Sigma_k \mid \\ &\quad \text{for } k = 1 \ldots, Q \end{aligned} \quad (25)$$

where μ_k is the estimated location of the centroid of class k; $\Sigma_k = [\sigma_{ij}^{(k)}]$ is the estimated $(n \times n)$ covariance matrix and $\mid \Sigma_k \mid$ is its determinant. The specimen is then assigned to the class with the lowest value of D_k.

The "learning" phase consists in the computation of μ and Σ from a set of examples $\{x_1, \ldots, x_N\}$ with known classes $\{c_1, \ldots, c_N\}$:

$$\mu_k = \frac{\sum_{r=1}^N x_r \delta(C(x_r) - k)}{\sum_{r=1}^N \delta(C(x_r) - k)} \quad (26)$$

$$\sigma_{ij}^{(k)} = \frac{\sum_{r=1}^N x_{ri} x_{rj} \delta(C(x_r) - k)}{\sum_{r=1}^N \delta(C(x_r) - k)} - \mu_{k_i}\mu_{k_j} \quad (27)$$

so that it takes only one pass through the data.

The performance of this kind of classifiers will be optimal, of course, only if the actual distribution of the data corresponds to a set of Gaussians centered at the class centroids; it is possible, however, to extend this idea for more complex cluster shapes, by approximating the data distribution for each class with a *sum* of Gaussian clouds. In this way, the classification is still done by taking the minimum of the discriminant functions (25), except that now k will be a 2–dimensional index: $k = (k_1, k_2)$ representing cluster k_2 of class k_1, and the minimization should be done over all clusters of all classes.

Note that the precise location of the clusters is not important, as long as the decision boundary between adjacent clusters of different classes is well approximated by a quadratic hypersurface; in many cases, this will happen provided the set of midpoints between the corresponding centroids samples the boundary manifold at a sufficiently high rate.

This suggests the following strategy for the learning phase of the compound classifier:

1 : Find a distribution of centers that samples the inter–class boundary manifold using the boundary–finding scheme described in section 2.2.2;

2 : Use the data points inside the Voronoi polytope of each center to learn the parameters of a local Gaussian classifier using (26) and (27) (computing the centroid parameters only for those classes that have representative points inside the polytope).

Note that the update rule (20) and the procedure for the adaptive determination of the number of centers guarantee that upon convergence, there will be observations for at least two different class clusters inside every Voronoi polytope.

The performance of this procedure is exemplified in figure 8, which illustrates a binary classification task for 2–dimensional data.

We also tested this technique with a classification problem in 5 dimensions. Table 1 compares its performance with other known classifiers (this table is included only as an illustration; the experiment was not intended to perform a formal comparison test). Note that since this procedure generates the same number of centers per class (in binary classification problems), its performance will deteriorate if the data are not evenly distributed among classes. In this case, its performance may become worse than that of other classifiers.

The training data set consisted of 1000 data points in the unit 5 dimensional cube $[0, 1]^5$. Of these points, $\frac{2}{3}$ were of class 1, and $\frac{1}{3}$ of class 2. Half of the points of class 1 were inside a 5 dimensional hypersphere of radius 0.07 and the other half are outside a 5 dimensional hypersphere of radius 0.2. The points of class 2 were between these 2 hyperspheres. The test set consisted of 10000 data points with the same distribution.

The rows labeled "RBF (adap)", "RBF (soft)" and "RBF (fixed) " correspond to the classifiers that are obtained by approximating the indicator function of one of the classes (say, class 2) with a linear combination of Gaussians with fixed covariance σI (in all cases we used $\sigma = 5.0$), and centered at a fixed set of points [22].

Classifier	training error (%)	test error (%)
LGC, 12 processors	11.1	13.7
LGC, 16 processors	10.7	12.9
NN, 20 hidden units	7.6	18.5
NN, 40 hidden units	2.7	15.7
NN, 50 hidden units	4.7	16.3
NN, 80 hidden units	2.5	15.9
RBF (adap), 38 centers	15.8	18.01
RBF (adap), 50 centers	15.5	17.32
RBF (adap), 64 centers	13.1	14.33
RBF (soft), 64 centers	15.6	16.95
RBF (fixed), 64 centers	19.8	21.21
RBF (adap), 78 centers	16.3	19.4

Table 1: Comparison of local Gaussian (LGC) and neural network (NN) classifiers on a 5 dimensional example.

These points may be found using: the k–means algorithm with a fixed number of centers ("RBF (fixed)" rows) or the adaptive strategy of section 2.2.1 ("RBF (adap)" rows). As one can see, the performance is significantly improved in the latter case, due to the fact that the center distribution is more uniform. A similar improvement has been reported if the standard k–means algorithm is replaced by the "soft–WTA" version (14) [23]. The performance of this scheme is included in the rows labeled "RBF (soft)", for comparison.

As one can see, the local Gaussian classifier has the best performance on the test set. The feedforward neural network with 50 hidden units and the RBF network with 50 centers have approximately the same number of parameters as the local Gaussian classifier with 16 processors.

In order to test the LGC on a set of real data we considered the same task of gender classification that has been considered by Brunelli and Poggio in [4]. Brunelli and Poggio had a data set consisting of 168 digitized pictures of frontal views of people without facial hair, and the task was the classification of the gender. There were 21 male and 21 female subjects in the data set, and 4 pictures per subject. Each picture was represented by a 16-dimensional vector of automatically extracted geometrical features, and we were provided with a set of 168 such vectors. In their analysis Brunelli and Poggio found that only few of the variables were relevant for the classification task, and therefore we used only the first 8 entries of each 16-dimensional vector, that included the most relevant variables found by Brunelli and Poggio. We did not attempt to find the best set of variables that describes this task. Since the number of training examples is small, relative to the dimensionality of the problem, we used the LGC without the boundary finding scheme for locating the centers. In order to test the performances of the LGC we adopted two procedures:

1. the data set was randomly split in 4 equal subsets, 3 of which were used for training and 1 for testing. This was repeated 10 times and the average training and testing errors computed;
2. the leave-one-out procedure described by Brunelli

and Poggio in [4].

In both cases the training and test error were less than 5% with 3 centers (3 Gaussians per class), and less than 8% with one Gaussian per class.

Figure 8 around here

3.3 Image Segmentation

As a final example, we consider an image segmentation problem that arises in the processing of certain biomedical images: scintigraphic images [11, 19], which are obtained by counting the number of radioactive particles that incide on each cell of a receptor array. The goal of the processing step is to obtain from these measurements an estimate of the radioisotope distribution in specific organs within the human body.

Particle count and radioisotope concentration are related by the Poisson distribution formula; therefore, the processing step consists in the restoration of a piecewise smooth function corrupted by Poisson noise. If it were possible to find the boundaries of the organ in question (e.g., the heart), the problem would reduce to filtering a smooth function within a given domain, for which effective methods are available (for example, Bayesian estimation methods with MRF priors and quadratic potentials to model the smoothness constraint [20]). In the example that we give here, we show that it is possible to adapt the methods that we have presented to classify the pixels of a scintigraphic image of the heart in such a way that one class corresponds approximately to the interior of the organ.

To do this, we will use the following concepts:

We assume that the two classes are characterized solely by the intensity level of the image, i.e., the interior (class 1) has high intensity with respect to the background (class 2). It is assumed that the classes are fuzzy sets [29] with membership functions of the form:

$$\begin{aligned} \phi_1(z) &= \frac{1}{1+\exp[-\beta(z-\theta)]} \qquad (28) \\ \phi_2(z) &= 1-\phi_1(z) \end{aligned}$$

where β and θ are positive parameters.

The formulae for the parameters of the discriminant functions of the local Gaussian classifier c are modified in the obvious way:

$$\mu_k^c = \frac{\sum_r x_r \phi_k(z(x_r))}{\sum_r \phi_k(z(x_r))} \qquad (29)$$

$$\sigma_{ij}^{(c,k)} = \frac{\sum_r x_{r_i} x_{r_j} \phi_k(z(x_r))}{\sum_r \phi_k(z(x_r))} - \mu_{k_i}^c \mu_{k_j}^c \qquad (30)$$

where the sums are taken over the learning domain of the classifier; x_r denotes the coordinates of pixel r of the image, and $z(x_r)$ denotes the value of the observed intensity.

The learning domain of each local classifier is taken as before, as the Voronoi polygon of a center that samples the image in an appropriate way. In this particular example, we are interested in segmenting the left ventricule of the hearth taken from a left anterior oblique projection. From this viewpoint, the ventricule appears as a high intensity "donut" over a dark background (see figure 9–a). Therefore, it is desirable that the centers are located uniformly along a closed, smooth curve that is attracted towards the higher intensity region of the image (note that the quadratic decision surface of each local classifier may be an hyperbola, and therefore, it can adequately segment a region that looks like a band within its domain).

Since a scintigraphic image is actually representing particle counts, we may use update rule (24) directly, considering that at each location x_r there are $z(x_r)$ data points. To get an appropriate behavior for this update rule, however, it is necessary that all the data points are visited in a random order (see lemma 1). To obtain this condition, it is not enough to visit the sites of the lattice randomly; it is also necessary, when, each site i is visited, to "flip a coin", and only update the corresponding center location with probability $P_{update} = z(x_i)/z_{max}$, where $z_{max} = \max_i z(x_i)$.

Figure 9 around here

The results of this procedure applied to the real scintigraphic image of figure 9–a are shown in figure 9–b. The white squares indicate the center locations, and the white line the final compound decision boundary. The threshold θ was obtained as the minimum between the two largest peaks of the global histogram of the image; in the experiments we performed, however, we found that the final results are not very sensitive to the precise value of neither this nor the other parameter (β).

4 Conclusions

In this paper we have analyzed the local K-Means algorithm, and have presented some extensions that increase its range of applicability. Our main contributions are the following:

i) We have established sufficient conditions for the convergence of this algorithm to a (local) minimum of a quadratic distorsion measure (lemma 1). In doing so, we showed that it can be obtained as the limit (as the parameter β becomes large) of a family of algorithms which are closely related to those obtained by minimizing an information distorsion measure. For moderate values of β, we showed that these algorithms can be used for unsupervised learning (clustering) tasks, since they can find a neighborhood structure for the centers that reflects the structure of the data.

ii) We showed that by varying the parameter β and adding a noise term to the update equation, it is possible to improve significantly the performance of the algorithm for clustered data.

iii) We introduced a modification to this algorithm that consists in augmenting the state of each processor (center) so that it keeps track of its own dynamic behavior. With this modification, it is possible to control the algorithm, so that, for example, the centers sample the inter–class boundary manifold (for multi–class data) directly. Since this manifold has one dimension less than the data themselves, and it contains the most relevant locations for classification purposes, this may represent a considerable saving in computational resources. We also showed that this augmented state may be used for other purposes, such as to control the number of centers in an automatic way.

iv) We showed that the self–organizing property of certain variations of the LKMA (specifically, Kohonen's self–organizing nets) can be put in a rigorous framework by considering the procedure as a Bayesian estimator with a Gibbsian (MRF) prior. This formulation has several advantages: firstly, the algorithm convergence can be rigorously established, and the properties of the stable states better understood; secondly, it suggests the use of multi–scale (pyramid) strategies that accelerate the convergence and conduct the algorithm to better final configurations, and finally, it permits the easy extension of the algorithm to generate a wide variety of organized configurations. Thus, we presented an application that consisted in finding the smooth curves that define the inter–class boundaries in 2–dimensional segmentation problems from sparse data.

v) We presented a classification scheme that combines the power of the Gaussian classifiers with the boundary–sampling properties of the extended LKMA, allowing the construction of very complex decision boundaries with very few computational elements. We exemplified the performance of these Local Gaussian Classifiers, both in "classical" classification problems and in fuzzy classification tasks related to image segmentation.

Other extensions of great interest are related to the use of these procedures for the optimal location of centers in function approximation problems. This is one of the subjects of our current research.

A Convergence of Stochastic Gradient Descent

In this appendix we prove that, under certain assumptions, a class of stochastic gradient descent techniques, that include "backpropagation" as a particular case, converges to the same result as the standard gradient descent algorithm. This fact is stated in lemma 2.1 (see section 2), which is an application of the following theorem [16]:

Theorem A.1 (Kushner and Clark, 1978) *Consider the following sequence in R^n:*

$$y_{n+1} = y_n + a_n h(y_n, \xi_n) + a_n \eta_n \qquad (31)$$

where $\{\xi_n\}$ is a sequence of random variables that do not depend on the $\{y_n\}$ and $\{\eta_n\}$ is a sequence of random variables converging to zero. Assume the following:

A1: the sequence $\{y_n\}$ is bounded with probability 1;

A2: $h(\cdot,\cdot)$ is a bounded measurable R^r-valued function;

A3: there are non-negative measurable real-valued functions $\theta(\cdot)$, $g(\cdot,\cdot)$ such that $\theta(\cdot)$ is nondecreasing as its argument increases, $\theta(u) \to 0$ as $u \to 0$, $\theta(\cdot)$ and $g(\cdot,\cdot)$ are bounded on bounded sets and

$$||h(y,\xi) - h(y',\xi)|| \leq \theta(||y-y'||) g(y,y') \quad ;$$

A4: there is a continuous function $\bar{h}(\cdot)$ such that for each $\epsilon > 0$ and each y:

$$\lim_{n\to\infty} P\{\sup_{m\geq n} ||\sum_{i=n}^{m} a_i [h(y,\xi_i) - \bar{h}(y)]|| \geq \epsilon\} = 0$$

A5: $\{a_n\}$ is a sequence of positive real numbers such that $a_n \to 0$ as $n \to 0$, and $\sum_{n=1}^{\infty} a_n = \infty$.

Let S be a locally asymptotically stable point of the ordinary differential equation:

$$\dot{y} = \bar{h}(y)$$

with domain of attraction A_S. Then, if $y_n \in G$ for all n, for some compact domain $G \subset A_S$, $\{y_n\}$ converges to S with probability one.

Lemma 2.1 can be derived from theorem (A.1) setting $\{\xi_n\}$ to be a sequence of random variables, with uniform probability distribution, that take values in $\{1,2,3,\ldots,N\}$ and setting

$$h(y,\xi) = -\nabla f_0(y) - \nabla f_\xi(y) \quad , \quad \bar{h}(y) = -\nabla F(y) \ .$$

Under these conditions we just need to check the validity of assumptions A3 and A4 in theorem (A.1).

A3: If we assume that the f_i are differentiable functions whose derivatives satisfy a Lipschiz condition, then

$$|| - \nabla f_0(y) - \nabla f_\xi(y) + \nabla f_0(y') + \nabla f_\xi(y')|| \leq M||y-y'||$$

for some positive number M. Therefore assumption A3 holds, with $\theta(u) = u$ and $g(y) = M$.

A4: We have to prove that for each $\epsilon > 0$ and for each y

$$\lim_{n\to\infty} P\{\sup_{m\geq n} ||\sum_{i=n}^{m} a_i [-\nabla f_0 - \nabla f_{\xi_i} + \nabla F]|| \geq \epsilon\} = 0 \quad . \qquad (32)$$

Defining the random variable

$$s_i = a_i [\nabla F - \nabla f_{\xi_i} - \nabla f_0]$$

condition (32) asserts that the sequence

$$c_n = \sup_{m\geq n} ||\sum_{i=n}^{m} s_i||$$

converges in probability. It is sufficient to show that the series $\sum_{i=n}^{m} s_i$ converges with probability one (Kushner and Clark, page 32). We use the following theorem by Kolmogorov and Khinchin (see [27], page 359):

Theorem A.2 *Let $\{z_n\}$ be a sequence of independent random variables with zero mean. Then, if*

$$\sum_{n=1}^{\infty} E[z_n^2] < \infty$$

the series $\sum_{n=1}^{\infty} z_n$ converges with probability 1.

In order to apply this theorem to the random variable s_i, we first need to check that s_i has zero mean. The probability distribution of the random variables ξ has been assumed to be uniform, and therefore $P(\xi_i) = \frac{1}{N}$. We then have:

$$\begin{aligned} E[s_i] &= a_i E[\nabla F - \nabla f_{\xi_i} - \nabla f_0] = \\ &= a_i[\nabla F - E[\nabla f_{\xi_i}] - \nabla f_0] = \\ &= a_i \nabla F - \frac{1}{N}\sum_{n=1}^{N} \nabla f_i - \nabla f_0] = 0 \ . \end{aligned}$$

Similarly we have

$$E[s_i^2] = a_i^2 E[(\nabla F - \nabla f_{\xi_i} - \nabla f_0)^2] = a_i^2 C(y)$$

for some function $C(y)$. If we assume that

$$\sum_{n=1}^{\infty} a_n^2 < \infty$$

then $\sum_{n=1}^{\infty} E[s_n^2] = C(x)\sum_{n=1}^{\infty} a_n^2 < +\infty$. Therefore the series $\sum_{i=1}^{n} s_i$ converges with probability 1, assumption A4 holds, and lemma 2.1 is proved.

Note that the assumptions about the boundedness of the sequence $\{y_n\}$, and the boundedness of the gradient of the f_i are not as restrictive as they seem, and in practice may be dropped. In fact, we can restrict the sequence $\{y_n\}$ to a compact region $G \subset R^r$, considering, instead of eq. (31), the following sequence:

$$y_{n+1} = \Pi_G[y_n - a_n(\nabla f_0(y_n) + \nabla f_{\xi_n}(y_n)) + a_n \eta_n] \quad (33)$$

where Π_G is the projection onto G. In this case, lemma 2.1 now holds if we substitute ∇F in eq. (12) with $\Pi_G \nabla F$. Choosing G sufficiently large to contain the fixed points of interest, the conclusion of the theorem is practically unchanged. Moreover, since now the sequence is restricted to the compact set G, the derivatives of the f_i are certainly bounded on G, being continuous, and therefore the assumpions of boundedness of the gradient of the f_i can be dropped.

Acknowledgements We are grateful to Tomaso Poggio and John Harris for useful discussions and for a critical reading of the manuscript. We also thank Roberto Brunelli for providing us the gender classification data set.

References

[1] M.R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[2] M. Benaim and L. Tomasini. Competitive and self-organizing algorithms based on the minimization of an information criterion. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 391–396. Elsevier Science Publishers B.V. (North-Holland), 1991.

[3] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Royal Statist. Soc.*, 36(B):192–326, 1974.

[4] R. Brunelli and T. Poggio. HyperBF networks for gender recognition. In *Proceedings Image Understanding Workshop.* Morgan Kaufmann, San Mateo, CA, January 1992.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[6] R. Durbin and G. Mitchison. A dimension reduction framework for understanding cortical maps. *Nature*, 343:644–647, February 1990.

[7] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[8] A. Gersho. On the structure of vector quantizers. *IEEE Transactions on Information Theory*, 28(2):157–166, 1982.

[9] A. Gersho and R.M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, Boston, 1991.

[10] McLachlan G.J. and Basford K.E. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, New York and Basel, 1988.

[11] T. Inouye. Computer processing of scintillation camera gas. *Nucl. Instrum. Meth.*, 124:215–219, 1975.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[13] R. Kinderman and J.L. Snell. *Markov Random Fields and their applications.* Amer. Math. Soc., Providence, RI, 1980.

[14] T. Kohonen. *Self-organization and associative memory.* Springer-Verlag, Berlin, 1984.

[15] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[16] H. Kushner and D. Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1978.

[17] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Proceedings of the IEEE*, Com-28(1):84–95, January 1980.

[18] J. MacQueen. Some methods of classification and analysis of multivariate observations. In L.M. LeCam and J. Neyman, editors, *Proc. 5th Berkeley Symposium on Math., Stat., and Prob.*, page 281. U. California Press, Berkeley, CA, 1967.

[19] J. Maeda and K. Murata. Digital restoration of scintigraphic images by a two-step procedure. *IEEE Trans. on Medical Imaging*, MI-6(4):320–324, December 1987.

[20] J. L. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *J. Amer. Stat. Assoc.*, 82:76–89, 1987.

[21] T. Martinez and K. Schulten. A "Neural Gas" Network Learns Topology. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier Science Publishers B.V. (North-Holland), 1991.

[22] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[23] S.J. Nowlan. Soft competitive adaptation. Ph.D. Thesis CMU-CS-91-126, Carnegie Mellon Univ., Pittsburgh, PA, April 1991.

[24] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[25] H. Ritter and K. Schulten. On the stationary state of kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54:99–106, 1986.

[26] I.A. Rozanov. *Markov random fields.* Springer-Verlag, New York, 1982.

[27] A.N. Shiriaev. *Probability*, volume 95 of *Graduate texts in mathematics.* Springer-Verlag, New York, 1984.

[28] H. White. Learning in artificial neural networks: a statistical perspective. *Neural Computation*, 1:425–464, 1989.

[29] L. Zadeh. *An Introduction to fuzzy logic applications in intelligent systems.* Kluwer Academic, Boston, 1992.

FIGURE CAPTIONS

Figure 1. Centers and neighborhood structures for 2–Dimensional data sets. The centers (circled dots in panels (c) and (d)) for the data sets shown in panels (a) and (b), respectively, were found using the adaptive algorithm described in section 2.2.1. The neighborhood structure is represented by lines that connect two centers if their average link status exceeds 0.1 (see text).

Figure 2. Representative centers for the 2–Dimensional data set of panel (a) found by the standard LKMA (panel(b)) and by the soft WTA algorithm (14) (panel(c)). The number of centers (9) was kept fixed in both cases.

Figure 3. Panel (a) portrays a 2–class, 2–dimensional data set of 200 points; points of class 1 are represented by dots, and points of class 2 by "+" signs or small circles. The circled dots of panel (b) represent the centers of the boundary set (see section 2.2.2). the upper threshold θ_s was set to 0.05 to have approximately 10 points per center.

Figure 4. "Folded" lattice obtained as a local minimum of energy (20) with quadratic potentials and cliques of size 3. (b) Lattice obtained with the pyramid process described in the text. The data points are uniformly distributed in the unit square.

Figure 5. "Pyramid" process for adding centers: circles represent the nodes of the initial 3×3 lattice; "X" signs correspond to the centers added after the first refinement.

Figure 6. Boundaries from sparse data: panel (b) portrays the discrete boundary curve (polygonal line) obtained as a fixed point of (23) for the data set of panel (a) (dots represent points of class 1 and plus signs or small circles, points of class 2) with $\lambda = 0.01$. Panel (c) portrays the fixed point for $\lambda = 0.1$.

Figure 7. Panels (c) and (d) portray boundary curves found by algorithm (23) for the data sets of panels (a) and (b) (dots represent points of class 1 and plus signs or small circles, points of class 2).

Figure 8. Decision boundaries found by the local Gaussian classifier described in section 3.2 for the data set of figure 3.

Figure 9. Panel (a) portrays a real scintigraphic image of a human heart taken from a left anterior oblique projection. Panel (b) shows the superimposed boundaries found by the fuzzy LGC described in section 3.3.