PARTIALLY SUPERVISED CLUSTERING FOR IMAGE SEGMENTATION

AMINE M. Bensaïd,† LAWRENCE O. HALL,‡§ JAMES C. BEZDEK¶ and LAURENCE P. CLARKE**

†Computer Science and Math Division, Al Akhawayn University in Ifrane (AUI), P.O. Box 104, Ave. Hassan II, Ifrane 53000, Morocco
‡Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, U.S.A.
§Department of Computer Science, University of West Florida, Pensacola, FL 32514, U.S.A.
¶Department of Radiology, University of South Florida, Tampa, FL 33620, U.S.A.

(Received 12 September 1994; in revised form 25 June 1994; received for publication 15 August 1995)

Abstract—All clustering algorithms process unlabeled data and, consequently, suffer from two problems: (P1) choosing and validating the correct number of clusters and (P2) insuring that algorithmic labels correspond to meaningful physical labels. Clustering algorithms such as hard and fuzzy c-means, based on optimizing sums of squared errors objective functions, suffer from a third problem: (P3) a tendency to recommend solutions that equalize cluster populations. The semi-supervised c-means algorithms introduced in this paper attempt to overcome these three problems for problem domains where a few data from each class can be labeled. Segmentation of magnetic resonance images is a problem of this type and we use it to illustrate the new algorithm. Our examples show that the semi-supervised approach provides MRI segmentations that are superior to ordinary fuzzy c-means and to the crisp k-nearest neighbor rule and further, that the new method ameliorates (P1)-(P3).

Cluster analysis Fuzzy c-means Image segmentation Magnetic resonance images

1. INTRODUCTION

This paper describes a new method for classification. The method is well suited to problems such as the segmentation of Magnetic Resonance Images (MRI), which is the domain discussed in this paper. Many techniques for segmentation of MR images were recently surveyed in reference (1). Here, we concentrate on a derivative of the c-means clustering algorithms that is designed to overcome three problems: (P1) choosing and validating the correct number of tissue types in the image; (P2) insuring that algorithmic labels correspond to meaningful physical labels of the tissue classes in the image; and (P3) a tendency of the c-means algorithms to stop at solutions that equalize cluster populations. All clustering algorithms suffer from (P1) and (P2); and (P3) is a well-known drawback of both hard c-means (HCM)(2) and Fuzzy c-means (FCM)(3) The way (P1) and (P2) are overcome in this paper is to assume that labeled data is available for every class. Hence, the algorithmic problems of determining the number of classes and their labels are side-stepped by the use of (at least some) labeled data from each class. The algorithm we propose is particularly well suited to image segmentation because a human can (and usually does anyway) examine the image and select a few clearly defined pixels from classes, which can subsequently be used to guide the modification we describe towards a realistic labeling of the remaining pixels. A small set of labeled pixels provides a clustering algorithm with a form of partial supervision. Our approach is developed in the context of segmentation of MRIs, but it can be used for data from any process, as long as a few data from each class can be labeled.

2. CLUSTERING

Object data are represented as \( X = \{x_1, x_2, \ldots, x_n\} \), \((n)\) feature vectors in feature space \( \mathcal{X} \). The \( j \)th object (some physical entity such as a person, airplane, seismic record, image pixel, etc.) has \( x_j \) as its numerical representation; \( x_{jk} \) is the \( k \)th characteristic (or feature) associated with object \( j \). Perhaps the most basic idea in pattern recognition is the class label. There are three types of labels extant in the literature: crisp (or "hard" or nonfuzzy), fuzzy and probabilistic. Let \( c \) denote the number of classes, \( 1 \leq c \leq n \), and define three sets of label vectors in \( \mathcal{X} \) as follows:

\[
\begin{align*}
N_{f_{w}} &= \{ y \in \mathcal{X} | y_i \in [0, 1] \forall i \} \\
&= \text{(unconstrained) fuzzy; } \quad (1a) \\
N_{fc} &= \{ y \in N_{f_{w}} | \sum_{i=1}^{n} y_i = 1 \} \\
&= \text{(constrained) fuzzy/prob.; } \quad (1b) \\
N_{c} &= \{ y \in N_{fc} | y_i \in \{0, 1\} \forall i \} \\
&= \text{crisp (or hard). } \quad (1c)
\end{align*}
\]
$N_c$ is the canonical (unit vector) basis of Euclidean $c$-space; $N_{fo}$ a piece of a hyperplane, is its convex hull; and $N_{fo}$ is the hypercube in $\mathbb{R}^c$. Figure 1 depicts these sets for $c = 3$. The $i$th vertex of $N_{c}, e_i = (0,0,\ldots, \frac{1}{c}, \ldots, 0)^T$, is the crisp label for class $i$, $1 \leq i \leq c$. The vector $y = (0.1, 0.6, 0.3)^T$ is a typical constrained label vector; its entries lie between 0 and 1, and sum to 1. The interpretation of $y$ depends on its origin. If $y$ is generated by, say, the fuzzy $c$-means clustering method, we call $y$ a fuzzy label. If $y$ originates from a method such as maximum likelihood estimation in mixture decomposition, $y$ would be a probabilistic label. The cube $N_{fo} = [0, 1]^3$ is termed unconstrained label vector space; vectors such as $z = (0.7, 0.2, 0.7)^T$ have each entry between 0 and 1, but are otherwise unrestricted. Labels in $N_{fo}$ are produced, for example, by feed-forward neural networks that have unipolar sigmoidal transfer functions at each of $c$ output nodes.

Clustering in unlabeled data set $X$ is the assignment of (hard or fuzzy or probabilistic) label vectors to the $\{x_k\}$ and hence, to the objects generating them. If the labels are hard, we hope they identify $c$ “natural subgroups” in $X$. Clustering is also called unsupervised learning, the word learning referring here to learning the correct labels (and possibly vector prototypes or quantifiers) for “good” subgroups in the data.

$\cdot$-partitions of $X$ are sets of $(cn)$ values $\{u_{ik}\}$ satisfying some or all of the conditions:

\begin{align}
0 \leq u_{ik} \leq 1 & \quad \forall i, k; \\
0 < \sum_{k=1}^{n} u_{ik} < n & \quad \forall i = 1, 2, \ldots, c; \\
\sum_{i=1}^{c} u_{ik} = 1 & \quad \forall k = 1, 2, \ldots, n.
\end{align}

Using equations (2) with the values $\{u_{ik}\}$ arrayed as a $(c \times n)$ matrix $U = [u_{ik}]$, we define:

\begin{align}
M_{fcon} &= \{U \in \mathbb{R}^{c \times n} | u_{ik} \text{ satisfies equations (2a) and (2b)} \forall i, k\}; \\
M_{fc} &= \{U \in \mathbb{R}^{c \times n} | u_{ik} \text{ satisfies equation (2c)} \forall i, k\}; \\
M_{c} &= \{U \in M_{fcon} | u_{ik} = 0 \text{ or } 1 \forall i, k\}.
\end{align}

Equations (3a), (3b) and (3c) define, respectively, the sets of unconstrained fuzzy, constrained fuzzy or probabilistic, and crisp $c$-partitions of $X$. Column $k$ of $U$ (denoted here as $U_{ak}$, in $M_{fcon}(M_{fc}, M_{c})$) is the label vector from $N_{fo}(N_{c}, N_{c})$ for object vector $x_k$. The reason these matrices are called partitions follows from the interpretation of $u_{ik}$ as the membership of $x_k$ in the $i$th partitioning subset (cluster) of $X$. Examples of matrices from each of these sets are:

\begin{align*}
U_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \in M_{cn}, \\
U_2 &= \begin{bmatrix} 1 & 0.25 & 0.15 \\ 0 & 0.15 & 0.10 \\ 0 & 0.60 & 0.75 \end{bmatrix} \in M_{fc}, \\
U_3 &= \begin{bmatrix} 1 & 0.65 & 0.1 \\ 0 & 0.65 & 0.75 \end{bmatrix} \in M_{fcon}.
\end{align*}

$U_1$ corresponds to the crisp clusters $\{x_1\}$ and $\{x_2, x_3\}$. $U_2$ shows constrained clusters; $U_{2,32} = 0.60$ indicates that $x_3$ is in class 3 to the degree 0.60; or, in the probabilistic context, that $x_3$ is (completely) in class 3 with probability 0.60. $M_{fcon}$ offers degrees of fuzzy membership that are not constrained to sum to 1 across all clusters, such as those in $U_2$. Fuzzy cluster-

![Fig. 1. Hard, fuzzy and probabilistic label vectors (for $c = 3$ classes).](image-url)
Partially supervised clustering began with Ruspini; Bezdek and Pal contains many seminal papers on this topic. Algorithms that produce unconstrained fuzzy partitions of $X$ are relatively new; see Krishnapuram et al.  for a recent approach.

A classifier is any function $D: P \rightarrow N_f$. The value of $D$ at any $z \in P$ is $y = D(z)$, the label vector for $z$ in $N_f$, $D$ is a crisp classifier if $D(z)$ is (some) $e_i$ in $N$ for every $z \in P$; otherwise, the classifier is fuzzy or probabilistic.

Since definite class assignments are usually the ultimate goal of classification and clustering, outputs of algorithms that produce label vectors in $N_{fc}$ or $N_{fc}'$ are usually transformed into crisp labels. Almost all non-crisp classifiers are converted to crisp ones using the maximum coordinate conversion function $H: N_f \rightarrow N_c$:

$$H(y) = e_i \Leftrightarrow \max_j \{y_j\} = y_{ip}$$

If $y = D(z)$, $H$ simply finds the crisp label vector $e_i$ in $N_c$ closest to $y$ [which, by equation (4), corresponds to the maximum coordinate of $y$] and assigns this label to $z$. The rationale for using $H$ depends on the algorithm that produces the label vector $y$. For example, the justification for using equation (4) for outputs from the $k$-nearest neighbor rule is simple majority voting. If $y$ is obtained from mixture decomposition, using $H$ is Bayes rule; label $z$ by its class of maximum a posteriori probability. Also, if the labels are fuzzy, this step is called defuzzification of $U$ by the maximum membership rule. We shall call each of these methods realized by $H$ hardening of $U$. Clustering algorithms produce partitions, which are sets of label vectors. For fuzzy partitions, the usual method of hardening is the application of equation (4) to each column $U_{ik}$ of $U$.

### 2.1 Maximum membership (MM) conversion

The crisp maximum membership partition $U_{MM}$ in $M_{fc}$, corresponding to any $U \in M_{fc}$ is:

$$U_{MM}(ik) = H(U_{ik}) = e_j \Leftrightarrow u_{jk} \geq u_{ik}, \quad j = 1, 2, \ldots, c, \quad j \neq i. \quad (5)$$

Choosing the maximum membership as the best pointer to a crisp label is a natural and plausible way to harden $U$, but is not the only way.

### 3. THE HARD AND FUZZY $c$-MEANS ALGORITHMS

The most widely used objective function for crisp clustering in $X$ is the classical within groups sum of squared errors objective function, defined as:

$$J_1(U, V: X) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} |x_k - v_i|^2.$$  

where $V = (v_1, v_2, \ldots, v_c)$ is a vector of (unknown) cluster centers (weights or prototypes), $v_i \in \mathbb{R}^p$ for $1 \leq i \leq c$, $U \in M_{fc}$ and $|*|$ is the Euclidean norm. Optimal partitions $U^*$ of $X$ are taken from pairs $(U^*, V^*)$ that are "local minimizers" of $J_1$. Approximate optimization of $J_1$, by the hard $c$-means (HCM) algorithm is based on iteration through the following necessary conditions for its local extremum:

**Hard $c$-means (HCM) theorem.**

$(U, V)$ may minimize $J_1$ only if:

$$u_{ik} = \begin{cases} 1; & |x_k - v_i| \leq |x_k - v_{ik}|, \quad j = 1, \ldots, c; j \neq i; \\ 0; & \text{otherwise} \end{cases} \quad (7a)$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik} x_k}{\sum_{k=1}^{n} u_{ik}}, \quad 1 \leq i \leq c. \quad (7b)$$

Ties in equation (7a) are resolved arbitrarily. In the context of MRI segmentation, HCM produces crisp partitions, so each pixel receives a unique tissue class assignment from this method. The well-known generalization of equation (6) to the infinite family called the fuzzy $c$-means functionals is:

$$J_m(U, V: X) = \frac{1}{m} \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m |x_k - v_i|^2, \quad (8)$$

where $m \in [1, \infty)$ is a weighting exponent on each fuzzy membership;

$U \in M_{fc}$ is a constrained fuzzy $c$-partition of $X$; $V = (v_1, v_2, \ldots, v_c)$, are $c$ vector prototypes in $\mathbb{R}^p$; $A$ is any positive definite $(p \times p)$ matrix;

Approximate optimization of $J_m$ by the fuzzy $c$-means (FCM) algorithm is based on iteration through the following necessary conditions for its local extremum:

**Fuzzy $c$-means (FCM) Theorem.**

Assume $m \geq 1$ and $|x_k - v_i|^2 > 0, \quad 1 \leq i \leq c, \quad 1 \leq k \leq n$. $(U, V)$ may minimize $J_m$ only if:

$$u_{ik} = \left[ \sum_{j=1}^{c} \left( \frac{|x_k - v_j|^2}{|x_k - v_i|^2} \right)^{2(m-1)} \right]^{-1}, \quad 1 \leq i \leq c; \quad 1 \leq k \leq n; \quad (10a)$$

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m}, \quad 1 \leq i \leq c. \quad (10b)$$

See reference (3) for a discussion of the case when one or more $|x_k - v_i|^2 = 0$. A brief specification of these two procedures follows.

**Hard/fuzzy $c$-means (HCM/FCM) algorithms**

1. Given unlabeled data $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^p$
2. Choose parameters $T, H, m \geq 1$ and $c > 0$.
3. Initialize $U_0 \in M_{fc}$ randomly.
4. Compute $(x_{i,0})$ with equations (7b) or (10b), for $1 \leq i \leq c$.
5. For $t = 1, 2, \ldots, T$. 


4. The Semi-Supervised Hard and Fuzzy c-Means algorithms

A salient point about any clustering algorithm, including HCM/FCM, is that when X is composed entirely of unlabeled data, the method is completely unsupervised. Once the integer c is selected (which is not a trivial consideration), partitions of X that are generated by any clustering algorithm have the following form:

\[ U = \begin{bmatrix} u_{11} & \cdots & u_{1k} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ u_{k1} & \cdots & u_{kk} & \cdots & u_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ u_{y1} & \cdots & u_{yk} & \cdots & u_{yn} \end{bmatrix} \]

All n label vectors are unknown at initialization; therefore, \( U_0 = U_{ran} \) is a random matrix in \( M_{kn} \). At the end of clustering, each row of the terminal partition \( U_f = U_{stop} \) corresponds to a physical class, but the algorithm has no way to know which row is which class—it simply knows the rows as 1, 2, \ldots, c. If the data are truly unlabeled, this problem can only be resolved by expert human intervention. All clustering algorithms, then, are at the mercy of these two problems (among others):

(P1) choosing c during initialization, and

(P2) assigning physical labels to the classes at termination.

At the other extreme, if we begin with known crisp labels for all n of the \( x_i \)'s, then \( U_0 = U_{csp} \) is crisp, known and fixed, and the construction of a classifier based on X becomes supervised learning. In this case, two of the major headaches of clustering—viz. (P1) and (P2)—are avoided by the use of labeled data from each class. However, supervision also requires expert intervention to obtain the labeled data and, moreover, it may be impossible to really label every vector in the data. For example, the data from a 256 x 256 MR image consist of 65,536 pixel vectors in \( \mathbb{R}^2 \). One reason for segmenting an image such as this automatically is to avoid the tedious, costly and practically impossible task of having a human assign a tissue class to each of these pixels.

The scheme we propose falls between these two extremes. We will avoid (P1) and (P2) by using some labeled data from each class and will continue to have the majority of the data in unlabeled form. This can be done by introducing a very few (hopefully) high quality labeled data from each class into X, thereby creating a partially labeled data set. This amounts to combining the ideas of supervised and unsupervised learning. Techniques of this kind in the context of c-means clustering were first discussed by Pedrycz, (7) who introduced two modifications to the FCM algorithm. First, a new summation term, corresponding to training patterns, was added to \( J_u \) [equation (8)]. Second, \( A \) [equation (9d)] became the covariance matrix of the training data rather than the identity matrix used here. The matrix \( A \) controls the shape of the clusters and since it uses training patterns only, it is crucial that the training set capture the shapes of clusters. We wish to apply the ssFCM algorithm to segmentation of magnetic resonance images, where the choice of training data that accurately describes cluster shape is difficult even for experts. On a single trial, it is not always possible for experts to choose acceptable training data for some fully supervised algorithms. (8) Hence, our approach is focussed on the kind of problem for which cluster shape may not be known and/or the training data may be imperfect.

We denote the partially labeled data, which provide the property of partial supervision to the algorithm, as:

\[ X = \left\{ x_1^1, \ldots, x_n^1, x_1^2, \ldots, x_n^2, \ldots, x_1^c, \ldots, x_n^c \right\} \]

\[ \text{labeled } 1 \text{ labeled } 2 \ldots \text{labeled } c \text{ unlabeled} \]

(12)

where superscripts show the class label and \( n_i \) is the number of data having label vector \( e_i \) for \( i = 1, 2, \ldots, c \). In this clash of notations, \( u \) indicates both the designation unlabeled when it is a superscript, as well as for the name of a membership function or value otherwise; and \( d \) indicates design (training) data. In the conventional approach to classifier design, \( X^d \) is used to train \( D \) (find its parameters); and then \( D \) is used to label \( X^u \). This is the approach we use with the k-nearest neighbor (k-nn) classifier described later.

If \( n_d = \sum n_i = n \) = \( |X| \) and \( n_u = |X^u| \), then \( n = |X| = n_d + n_u \). Portions of X induced by equation (12) have the form:

\[ U = \begin{bmatrix} U_{11}^{(1)} & \cdots & U_{1l}^{(1)} & \cdots & U_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ U_{l1}^{(1)} & \cdots & U_{ll}^{(1)} & \cdots & U_{ln}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ U_{n1}^{(1)} & \cdots & U_{nl}^{(1)} & \cdots & U_{nn}^{(1)} \end{bmatrix} \]

or

\[ U = \begin{bmatrix} e_1 & u_{11}^u & \cdots & u_{1l}^u & \cdots & u_{1n}^u \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ e_l & u_{l1}^u & \cdots & u_{ll}^u & \cdots & u_{ln}^u \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ e_n & u_{n1}^u & \cdots & u_{nl}^u & \cdots & u_{nn}^u \end{bmatrix} \]

(13b)

It will be cumbersome to continue using the notation in equation (13), so we may represent this matrix as the
In general, the \( n_s \) need not be equal, nor is it necessary that the column vectors in \( U^d \) be crisp.

In the sequel, \( U^d \) is fixed, so the initial matrix for c-means iteration has the form \( U_0 = [U^d|U^\nu] \) and the terminal (final) matrix from either HCM or FCM will have form \( U_f = [U^d|U^\nu]^T \). Ordinarily, once \( U_0 \) in (cM3) is determined, the next step in FCM would be to compute cluster centers \( \{v_{i,0}\} \) using equation (10b), which uses all \( n \) columns of \( U_0 \). However, for \( U_0 = [U^d|U^\nu] \), the last \( n_\nu \) columns of \( U_0 \) are initialized randomly subject to the constraints in equations (2a)-(c), whereas its first \( n_d \) columns are known to correspond to correctly labeled data, so it is advantageous to compute the first set of cluster centers using only the columns in \( U^d \). This results in an initial set of cluster centers that are "well-seeded" in the sense that they are influenced only by the data known to belong to their respective classes. This is done by computing \( v_{i,0} \) as the sample mean of the vectors labeled \( x_i \):

\[
v_{i,0} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_i^d, \quad 1 \leq i \leq c.
\]

Returning to equation (10a) equipped with these first-pass centers, FCM would ordinarily next calculate \( U_0 \) using the \( \{v_{i,0}\} \) just found, to update all \( n \) columns of \( U_0 \). However, the point of having labeled data is that, for the first \( n_d \) points, the labels are presumed correct. Consequently, we subsequently update only the \( n_\nu \) columns in \( U_\nu \), by calculating:

\[
u_{i,k} = \frac{1}{\sum_{j=1}^{c} \|x_i^d - v_{i,j}\|^m} \sum_{j=1}^{c} \left( \frac{\|x_i^d - v_{i,j}\|^m}{\|x_i^d - v_{i,j}\|^m + \sum_{k=1}^{n_d} (w_{i,k})^m} \right) x_i^\nu, \quad 1 \leq i \leq c, \quad 1 \leq k \leq n_\nu; \quad t = 1, 2, \ldots, T.
\]

Finally, once the initial cluster centers \( \{v_{i,0}\} \) have gotten the calculations off to a good start via equation (14b), we let them migrate in feature space, by returning to equation (10b) for subsequent updates of the \( \{v_i\} \).

The ideas embodied in equations (14a)–(d) almost capture the improvements we seek. The utility of equations (14a)–(d) hinges on having enough training data. If, for example, half of \( X \) were labeled, the training labels in \( U^d \) would in all likelihood have the desired effect, which is to equip the algorithm with \( c \) physically correct labels and enough good examples to enable it to make high quality estimates of the remaining labels for data set \( X^\nu \).

However, it is often the case in practice that \( n_\nu \) is quite small compared to \( n_d (n_\nu << n_d) \). For example, in a square image of size 256, there are 65 536 pixels. It is common to select and label subsets of, say 50–100 pixels/class from such an image. For a typical MR image, say, \( c = 7 \), so if 100 pixels/tissue class are labeled, \( n_\nu = 700 \), and \( n_d = 64 836 \). In this case, the training data comprise about 1.07% of the data and their influence on subsequent estimates of the cluster centers, as computed by equation (14d), can be overwhelmed by the remaining unlabeled samples.

Moreover, the numbers of pixel vectors in an MR image corresponding to each tissue class can be vastly different. For example, a small tumor (an important class to segment correctly) can consist of several hundred pixels, whereas gray matter can occupy thousands of pixels. Objective functions such as \( J_1 \) and \( J_\nu \) that minimize sums of squared errors are well known for their propensity to find solutions that "balance" the number of members in each cluster [reference (2) p. 220]. This tendency is a third serious problem when segmenting an MR image with HCM/FCM:

(P3) least squares objective functionals try to equalize cluster populations.

The reason HCM/FCM suffer from (P3) is that approximately-equal cluster populations result in smaller values of \( J_1 \) or \( J_\nu \). When equations (14a)–(d) are used, (P3) persists if \( n_d << n_\nu \). To counter the possible effect of unequal tissue populations in MR images, we introduce one more modification of c-means by weighting the few samples that are labeled more heavily than their unlabeled counterparts. This is done by introducing weights \( w = (w_1, w_2, \ldots, w_n) \) in equation (14d) as follows:

\[
v_{i,t} = \frac{1}{\sum_{k=1}^{n_d} w_k (x_i^d - v_{i,k})^m + \sum_{k=1}^{n_\nu} (w_{i,k})^m (x_i^\nu - v_{i,k})} \sum_{k=1}^{n_d} w_k (x_i^d - v_{i,k})^m + \sum_{k=1}^{n_\nu} (w_{i,k})^m (x_i^\nu - v_{i,k})^m, \quad 1 \leq i \leq c; \quad t = 1, 2, \ldots, T.
\]

In the most general formulation, \( w = (w_1, w_2, \ldots, w_n) \) is simply a vector of positive real numbers. If \( w_k \) is an integer \( > 1 \) for all \( k \), equation (14e) effectively uses \( w_k \) copies of labeled data point \( x_i^d \) each time the cluster centers are recomputed with equation (14e). There is, however, no mathematical reason to restrict \( w_k \) to just integers, nor is it necessary that \( w_k \) be greater than 1. These weights allow us to tailor the computation to agree with any expert knowledge that might be available about the quality or importance of each \( x_i^d \) as a training example. In the absence of such knowledge, the most rational approach is to take \( w_k = w \) for all \( k \). We summarize this discussion by formalizing the developed algorithm.

Semi-supervised fuzzy c-means (ssFCM) algorithms

ssFCM1. Given partially labeled data \( X = X^d \cup X^\nu \) as in equation (12), \( n_d = |X^d| \), \( n_\nu = |X^\nu| \), and \( n = |X| = n_d + n_\nu \). Note that \( c \) is known and fixed by the training data.

ssFCM2. Choose parameters \( w, T, \|x\|, m > 1 \), and \( \varepsilon > 0 \).
ssFCM3. Initialize $U_0 = [U^d | U^w]$, with $U^w_0 \in M_{fcw}$.

ssFCM4. Compute

$$v_{i,0} = \frac{\sum_{k=1}^{n_c} (u^w_{ik,0})^m x^d_k}{\sum_{k=1}^{n_c} (u^w_{ik,0})^m}, \quad 1 \leq i \leq c.$$  \hspace{1cm} (14b)

ssFCM5. For $t = 1, 2, \ldots, T$:

(a) Compute

$$u^w_{ik,t} = \left[ \sum_{j=1}^{c} \left( \frac{\| x^d_k - v_{j,t-1} \|}{\| x^d_k - u^w_{ik,t-1} \|} \right)^{2(m-1)} \right]^{-1},$$

$$1 \leq i \leq c; \quad 1 \leq k \leq n_w; \quad t = 1, 2, \ldots, T.$$  \hspace{1cm} (14c)

(b) Compute

$$E_t = \| U^w_t - U^w_{t-1} \| = \sqrt{\sum_{i=1}^{c} \sum_{k=1}^{n_w} (u^w_{ik,t} - u^w_{ik,t-1})^2}.$$  \hspace{1cm} (14d)

(c) if $E_t \leq \varepsilon$ stop, else compute

$$v_{i,t} = \left( \frac{\sum_{k=1}^{n_w} v_{i,t-1}}{\sum_{k=1}^{n_w} u^w_{ik,t}} \right)^m x^d_k + \sum_{k=1}^{n_w} u^w_{ik,t} x^d_k,$$

$$1 \leq i \leq c; \quad t = 1, 2, \ldots, T.$$  \hspace{1cm} (14e)

Next $t$.

If $n_d = 0$ (i.e. $n_w = n$) or $w = 0$, then ssFCM reduces to FCM. However, while we call this algorithm ssFCM, it does not bear a direct relationship to FCM, because the iteration in ssFCM between equations (14c) and (14e) is not necessary for the minimization of $J_w$. That is, ssFCM is a relative of FCM, but not a true generalization of it. Instead, ssFCM is a new clustering algorithm motivated by FCM. Another point: we did not carry along the development of the special case of ssFCM for $m = 1$, i.e. the generalization of HCM to ssHCM. However, for the record, ssHCM is the special case of ssFCM obtained by taking $m = 1$, and using equations (17a) and (b) instead of (10a)–(b) for (14c) and (e). Since we did not test ssHCM, it will not be further discussed here.

5. A SIMPLE EXAMPLE

To see how ssFCM mitigates the tendency of $J_w$ to equalize populations, we give a numerical example. Problem (P3) was illustrated (for HCM) in reference (2) using a data set shown in Fig. 2 and listed in Appendix A. This data set consists of 43 two-dimensional (2-D) vectors falling into one of two natural subgroups; the larger group is made up of 40 points and the smaller group contains three points. Figure 2(a) depicts FCM's partitioning of the data using $c = 2$, $m = 2$, $\varepsilon = 0.0001$ and the Euclidean norm. Figure 2(a) illustrates FCM's tendency to take away patterns from larger clusters and assign them to smaller clusters, a manifestation of (P3). This is a result of the location of the final cluster centers. In order to minimize the sum of squared distances from points to the center of their respective clusters, the center ($v_i$) of any small cluster drifts away from its natural location (the centroid of the smaller cluster) towards an adjacent (in feature space) bigger cluster; this phenomenon can be seen in Fig. 2(a). As a result, some elements that actually belong to the larger cluster become closer to $v_i$, and, therefore, get assigned by equation (5) to the smaller cluster; thus problem (P3).

In ssFCM, centers of smaller clusters can be prevented from migrating towards larger clusters; this is achieved by giving "high" weights to the training data corresponding to smaller clusters (the weights given to training patterns for larger clusters are not as critical). Figure 2(b) shows the partition and final cluster centers.

---

Fig. 2. Two clusters of unequal size. (a) Typically poor FCM clusters. (b) ssFCM finds the desired clusters.
obtained by ssFCM, run with the same parameters as FCM and with $w_j = 6 V_j$ as the weights for all training patterns; four training patterns from the larger class and one from the smaller class were used. Table 1 shows the number of patterns from the larger class assigned to the smaller class, with uniform weighting of training patterns from 1 to 10; ssFCM makes no errors for weights 4 and above. FCM makes 14 errors, as shown in Fig. 2(a); HCM makes 13 errors, as reported in reference (2). In Table 2, we show the results from an example in which the training set consists of just one pattern from one class, the smaller class. Again, with a weight 5 and above ssFCM makes no classification errors.

In order to provide some insight into how ssFCM differs from other fully supervised algorithms, a feed-forward back-propagation neural network (10) was tested on the same data set. The same five patterns were used for training with one from the small class and four from the larger class. The patterns from the larger class were those nearest the centroid of the data making up that class. For a 2 input 1 hidden unit 1 output (2-1-1) network the average error over 20 trials with different random initial weights was 3.5 misclassifications. In most cases the network simply converged to a solution with one training pattern misclassified, where all patterns were classified as belonging to the larger class. With a 2-2-1 network over 20 trials with different random initial weights, the average error was nine misclassifications. With three hidden units, a 2-3-1 architecture, the average error was six misclassifications. Over 20 trials with different initial weights, the average error was nine misclassifications. Each feature vector is assigned to the nearest-prototype as dictated by equations (14e) and (5), so if the cluster centers are well separated by the weighted training data, the number of feature vectors in each class can vary widely without adversely affecting the resulting cluster centers [determined by equation (14e)]. Hence, given a good choice of training data, the locations of the prototypes can be influenced enough to overcome problem (P3).

How are the weights to be chosen in equation (14e)? There are several possibilities and certainly more scenarios for training data than those explored here. Only the types of training data for which ssFCM appears most useful are discussed.

(1) The training data may be relatively plentiful, but their labels are not very certain.
(2) The number of training data may be small and their reliability unknown.
(3) There may be only a small amount of reliable training data relative to the available data.

In the first case, all weights should be chosen so that they are relatively small. If there are classes known or expected to be difficult to separate, then more weight can be given to the training data that describe them. In fact, if some patterns within the training data are considered good examples of the class these may be given more weight than the others.

The second case calls for a choice of weights as in the first case. The only exception would be that these weights should be just large enough to have some influence on the cluster centers. However, they should not dominate the placement of the final cluster centers and, hence, the final partition. Again, it can be useful to weight the "good" training vectors from a hard-to-separate class enough that they have a distinct influence on the location of the cluster center.

In the third case, the objective is to overcome (P3) and have some influence on all of the final clusters. This means that training data for hard to separate small clusters should be given enough weight so that the centroid of the cluster is strongly influenced by the training data. It also means that the weights for the training data need to be high enough that they have some influence in relation to the projected size of the class to which they belong.

### Table 1. Nearest prototype error rates for ssFCM, Example 1.

<table>
<thead>
<tr>
<th>$w_j$ (for all j)</th>
<th>Number of patterns from larger cluster assigned to smaller cluster by $U_{ss}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 2. Nearest prototype error rates for ssFCM, from Example 1 and train set consisting of just 1 example of the small class and 0 of the larger class.

<table>
<thead>
<tr>
<th>$w_j$ (for all j)</th>
<th>Number of patterns from larger cluster assigned to smaller cluster by $U_{ss}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

5.1. Some guidelines on setting weights

Let us examine the effect of our modification in equation 14(e) on the resulting cluster centers ($v_i$), that cause problem (P3). If we give strong weight to the training patterns from the smaller cluster, the center for that cluster will be drawn towards the centroid of the training patterns. As this weight approaches infinity, the cluster center ($v_i$) for the ith clusters will approach the centroid of the training patterns. If the training data and their associated weights make the cluster centers for two neighboring clusters (in feature space) sufficiently separate, the resulting number of nonfixed (test) feature vectors in the two clusters can be quite different. Each feature vector is assigned to the nearest-prototype as dictated by equations (14e) and (5), so if the cluster centers are well separated by the weighted training data, the number of feature vectors in each class can vary widely without adversely affecting the resulting cluster centers [determined by equation (14e)]. Hence, given a good choice of training data, the locations of the prototypes can be influenced enough to overcome problem (P3).

How are the weights to be chosen in equation (14e)? There are several possibilities and certainly more scenarios for training data than those explored here. Only the types of training data for which ssFCM appears most useful are discussed.

(1) The training data may be relatively plentiful, but their labels are not very certain.
(2) The number of training data may be small and their reliability unknown.
(3) There may be only a small amount of reliable training data relative to the available data.

In the first case, all weights should be chosen so that they are relatively small. If there are classes known or expected to be difficult to separate, then more weight can be given to the training data that describe them. In fact, if some patterns within the training data are considered good examples of the class these may be given more weight than the others.

The second case calls for a choice of weights as in the first case. The only exception would be that these weights should be just large enough to have some influence on the cluster centers. However, they should not dominate the placement of the final cluster centers and, hence, the final partition. Again, it can be useful to weight the "good" training vectors from a hard-to-separate class enough that they have a distinct influence on the location of the cluster center.

In the third case, the objective is to overcome (P3) and have some influence on all of the final clusters. This means that training data for hard to separate small clusters should be given enough weight so that the centroid of the cluster is strongly influenced by the training data. It also means that the weights for the training data need to be high enough that they have some influence in relation to the projected size of the class to which they belong.
In every case the choice of training data needs to be "near" the centroid of the class from which it is chosen or the weights can have unexpected effects. For example, Fig. 3 shows a two-class problem. If the two circled points are chosen for training and the training weights are chosen to be 1, all points are correctly classified. They are classified as belonging to a cluster centered to the left of $x = 0$ or right of $x = 0$ when their $x$ value is $<0$ or $>0$ respectively. However, the chosen training points are not so close to the respective centroids of the clusters. With $w_1 = w_2 = 10$ (the weights for the training patterns from the two classes equal 10), the point at $(0.4, 0)$ is misclassified. This example illustrates that even relatively low weights given to training patterns that are not near the class centroid can cause errors in the final ssFCM partitions that would not otherwise occur. However, this is the type of problem that all supervised classification systems suffer from when the training data is not "good" by the definition required by the particular algorithm.

6. SEGMENTATION OF MR IMAGES

In this section we discuss the application of FCM, ssFCM and the $k$ nearest neighbor ($k$-nn) algorithm\textsuperscript{(11)} to the segmentation of Magnetic Resonance Images. The results with the $k$-nn rule are provided, because in extensive studies\textsuperscript{(8)} with $k = 7$, as used in the work reported here, this method has proved to provide the best segmentations of a number of supervised techniques. The general idea of segmenting MR images of the human brain has been discussed extensively elsewhere,\textsuperscript{(1)} so we review here only the construction of the feature data. Input data to the classifier or clustering scheme will typically be some subset of pixel intensities of sensed quantities, or more generally, a vector of features constructed from them, at each spatial location in the image. Figure 4 illustrates the general procedure for $p$-dimensional data.

The experiments in this section used $p = $ three-dimensional data sets constructed in this way from 10
Table 3. Tissue types and their artificial colors

<table>
<thead>
<tr>
<th>Tissue no. (i)</th>
<th>Tissue type</th>
<th>Color</th>
<th>( n_i ) for Fig. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Air</td>
<td>Blue</td>
<td>62</td>
</tr>
<tr>
<td>2</td>
<td>CSF</td>
<td>Yellow</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>White Matter</td>
<td>Red</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>Gray Matter</td>
<td>Orange</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Tumor</td>
<td>Gray</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>Necrosis</td>
<td>Purple</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Fat</td>
<td>Pink</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>Bone</td>
<td>Brown</td>
<td>39</td>
</tr>
</tbody>
</table>

different pathological patients. The representative image shown and all others were transverse images of 5 mm thickness obtained with standard spin echo techniques. Let \( T_{1ij}, T_{2ij}, \rho_{ij} \) denote the spin lattice relaxation (pulse repetition time \( TR = 600\, \text{ms} \), echo time \( TE = 20\, \text{ms} \)), transverse relaxation and proton density (\( TR = 3000\, \text{ms}, TE = 20\) and \( 80\, \text{ms} \), respectively) of pixel \((i,j)\) in an MRI slice of overall dimensions \((q \times r)\). We aggregate these into the pixel vector \( x_{ij} = (T_{1ij}, T_{2ij}, \rho_{ij}) \in \mathbb{R}^3 \); and put \( X = \{x_{11}, x_{12}, \ldots, x_{qr}\} \). Our images were \( 256 \times 256 \), resulting in sets of \( n = 65\,536 \) pixel vectors in \( \mathbb{R}^3 \). The 10 images generated 10 such data sets, \( X_1, X_2, \ldots, X_{10} \). There were \( c = 8 \) tissues of interest. Their names and colors are shown in the first two columns of Table 3.

The five views in Fig. 5 show outputs of several approaches to segmentation that are typical of images taken from all 10 of these patients. In each image, the classifications have been hardened by assigning each pixel to the class in which it has maximum membership using equation (5). Table 4 shows the computing protocols for each algorithm that is responsible for one of the five views in Fig. 5. All three algorithms used the Euclidean norm as the measure of distance. The number of training pixel vectors per class for ssFCM and the \( k\)-nn rule are shown in the last column of Table 3. The same training data were used for views 5(c), 5(d) and 5(E). Training data were selected interactively with a mouse technique, so the number of pixels per class varies because some tissue regions were much easier to select from than others. The mouse technique is to use a mouse to outline pixel regions from a given class in the raw T1 - T2- or PD-weighted image. The pixels outlined are (under program control) labeled as a specific user defined class and stored for use in ssFCM training sessions.

Comments on Figs 5(a) and (b)

Figure 5(a) shows the result of FCM with \( c = 8 \). This segmentation erroneously merges necrosis, the dis-shaped dark strip in the top, middle of the T1 image which is surrounded by white, into the CSF class (yellow). This has been a typical result of FCM on the images used in our experiments; with \( c = 8 \), FCM consistently misses one or more pathological tissues: tumor, edema (not applicable in Fig. 5) or necrosis, and instead finds several clusters (often at the boundary of the brain) deemed diagnostically insignificant by radiologists. This is NOT a manifestation of problem (P1), because we have supplied FCM with the (operator chosen) "correct" number of clusters, \( c = 8 \). It is more likely due to (P3), the problem of least-squares-based algorithms under greatly unequal populations in the data. In some cases, increasing the number of clusters to \( c = 10 \) results in recognizing one or more of the important clusters missed with \( c = 8 \). However, this increase can result in splitting clusters that were correctly recognized with \( c = 8 \) and/or in segmenting extra-cranial regions even more finely, while still missing the tumor, edema and/or necrosis tissue classes; Fig. 5(b) is a representative FCM segmentation with \( c = 10 \). Furthermore, the regions identified by FCM often had the wrong physical labels (problem P2), so the tissue classes seen in Figs 5(a) and (b), with \( c = 8 \) and \( c = 10 \), respectively, were colored manually after segmentation so that the eight tissue types itemized in Table 3 can be seen and compared visually with the other three segmentations. Figures 5(a) and (b) show how serious problems (P1)-(P3) can be.

The segmentation in Fig. 5(a) underestimates gray matter in favor of white matter. Also, necrosis is placed in the CSF class. Tumor is recognized, but the overall segmentation can be improved. In Fig. 5(b) gray matter is split into two classes, and necrosis remains in the CSF class. Also, the extra-cranial region is segmented a little finer by the addition of another class. Here, there was no improvement by overclustering.

Comments on Figs 5(c)-(d)

Figures 5(c) and (d) are segmentations of the same data made by ssFCM using the pixel vectors implied by column three of Table 3 for partial supervision. Since an operator selects and labels subsets of each of the eight tissue types shown in Table 3, ssFCM has (i) the correct (in the view of the operator selecting the training data) number of tissue types, and (ii) correct labels (colors) for the examples in each class. In other words, problems (P1) and (P2) are avoided by having labels for data from each operator-chosen class. Views in Figs 5(c) and (d) differ only in the weights used in equation (14e). In Fig. 5(c), \( w_j = 6 \) \( \forall j \) and in Fig. 5(d) this changes to \( w_j = 30 \) \( \forall j \).

The radiology researchers who evaluated Fig. 5 chose 5(c) as superior to 5(d), because it better differentiates tissues at a finer level. The differentiation between gray matter and tumor is best here. Necrosis is correctly recognized. The image in Fig. 5(d) also provides a good segmentation, but it is not as accurate in its separation of tissues.

These results suggest that the training data may not be perfectly accurate. A little weight provides a better segmentation than a greater weight. If this is indeed the case, we would expect that the segmentation by \( k\)-nn would not be too good, as it depends solely on the training data.
Fig. 5. Five segmentations of a tumor patient. The raw images are shown in the top row.

Table 4. Computational protocols for Figs 5(a), (b), (c), (d) and (e)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Algorithm</th>
<th>Type</th>
<th>k</th>
<th>c</th>
<th>m</th>
<th>(\varepsilon)</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(a)</td>
<td>FCM</td>
<td>Unsup.</td>
<td>na</td>
<td>8</td>
<td>2</td>
<td>0.01</td>
<td>na</td>
</tr>
<tr>
<td>5(b)</td>
<td>FCM</td>
<td>Unsup.</td>
<td>na</td>
<td>10</td>
<td>2</td>
<td>0.01</td>
<td>na</td>
</tr>
<tr>
<td>5(c)</td>
<td>ssFCM</td>
<td>Supv.</td>
<td>na</td>
<td>8</td>
<td>2</td>
<td>0.01</td>
<td>(w_j = 6 \forall j)</td>
</tr>
<tr>
<td>5(d)</td>
<td>ssFCM</td>
<td>Supv.</td>
<td>na</td>
<td>8</td>
<td>2</td>
<td>0.01</td>
<td>(w_j = 30 \forall j)</td>
</tr>
<tr>
<td>5(e)</td>
<td>k-nn rule</td>
<td>Supv.</td>
<td>7</td>
<td>8</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
</tbody>
</table>
Comments on Fig 5(e)

The segmentation in Fig. 5(e) is carried out with the k-nearest neighbor rule. We have processed many images with this algorithm in the last two years, and extensive experimentation with k indicates that k = 7 provides the best results.

The k-nn segmentation does differentiate tumor and necrosis. However, it does not differentiate between tumor and gray matter as well as ssFCM. k-nn grossly overestimates the amount of gray matter; the extra pixels identified as gray matter are actually CSF for the most part. Overall it does not provide a very good segmentation. This indicates that the training set, carefully chosen by an expert, was not actually very good (for k-nn). It is very difficult to correctly choose pixels that clearly belong to each class, as is apparently the case here.

6.1. Further results from MR imaging

A series of experiments to measure tumor volume from MR brain images using ssFCM, k-nearest neighbors (kNN), and a seed growing approach has been carried out. The study was to determine how sensitive the methods were to the choice of training data chosen by observers (technicians with some medical physics training). It is difficult to choose training data from the raw images such that the measured tumor volume is consistent over multiple trials. Four patient cases were used in the study with repeat scans available for two patients (three for patient 1 and five for patient 2). The results reported are over the ten tumor volumes obtained from the four patient cases. The tumors involved were either meningioma or glioblastoma multiforme. All patients were undergoing therapy during the course of repeat scans.

The results of the experiments are summarized in Table 5, in terms of the tumor volume variability resulting from the choice of the training data for each trial. A value w = 100 was used in ssFCM for all classes in this experiment. The differences in volume for multiple training sets chosen by one observer are given in the first row of Table 5 for two observers. ssFCM and the seed-growing method had the lowest variability intraobserver. The ssFCM algorithm had the lowest variability intra-observer as shown by the second column in Table 5. This experiment indicates that ssFCM is less sensitive to the choice of training data.

6.2. Initialization versus training: their influence on the ssFCM algorithm

As a partially supervised algorithm, ssFCM falls into a class of algorithms that is intermediate between the fully supervised and the fully unsupervised classes of learning algorithms. Partitions created by iterative clustering algorithms are sensitive to initialization. Specifically, the local minimum of J_n to which FCM will converge depends on the initial guess. In the MR1 domain, segmentations can be quite different for different initializations. In equation (14b) ssFCM is initialized with the training data. If the user of the algorithm is unsure of the "goodness" of the training data, we may revise (14b) to take form (14e):

\[ v_{i,0} = \left( \frac{\sum_{k=1}^{m} w_k (u_{ik})^w x_i^k}{\sum_{k=1}^{m} w_k (u_{ik})^w} \right) \]

for \( i = 1, 2, \ldots, c \); \( w_k > 0 \), (15)

where the \( u_{ik}^w \) are assigned subject to the constraints in equation (2). They may be taken from the end results of a previous clustering, created randomly or chosen in some other fashion. The initialization is now not governed solely by the training data.

One aspect of equation (14b) is that with good training data it provides an initialization of the cluster centers that will be relatively near the actual centroids of the clusters in feature space. Given this kind of initialization, FCM (we have found in practice) is more likely to converge to a final partition that is acceptable to the user. We have shown that in the MR1 domain, FCM is very sensitive to initialization, both in terms of the time it takes to converge and the quality of the final partition.

A question of interest is: can bad training data, which will provide a relatively poor initialization in equation (14b), be overcome? This is possible and was shown in Section 6, where the (partially supervised and) supervised methods produced poorer segmentation as more weight was given to the training data. Indeed, ssFCM with training weights \( w_j = 6 \forall j \) [Fig. 5(e)] produced the best results; the segmentation was not as good when the weights were increased to \( w_j = 30 \forall j \) [Fig. 5(d)]. Finally, quality deteriorated in Fig. 5(e) (with k-nn), where the segmentation was based fully on the training data. This deterioration is probably due to bad training data since k-nn's segmentation of MRIs is usually quite good. This suggests that bad training data were overcome by ssFCM using lower

---

Table 5. Inter- and intra-observer brain tumor variability among three supervised segmentation methods

<table>
<thead>
<tr>
<th></th>
<th>Variability for ssFCM (%)</th>
<th>Seed growing (%)</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-observer</td>
<td>6</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Inter-observer</td>
<td>4</td>
<td>17</td>
<td>5</td>
</tr>
</tbody>
</table>
training weights. Of course, if there is a lot of poor training data, or if the training data are given high weights in equation (14e), they will strongly influence the cluster centers; this is likely to result in a poor final partition of the data.

Another question involves whether a poor initialization when using equation (15) can be overcome, resulting in a final partition that may be judged as good. To provide some guidance on this question, ssFCM was run with an initial V created from the pixels of interest [represented in a \( U^d \) matrix, as in equation (14a)] selected from an MR image that was:

- of a different normal subject than the one that we ran ssFCM on;
- obtained by a different MRI acquisition protocol (fast-spin-echo) than the protocol (spin-echo) used to acquire the data of the normal subject ssFCM was run on. A fast-spin-echo data set has pixel intensities which are generally more than a factor of 2 times larger than those acquired using a spin-echo technique.

The initialization is guaranteed to be poor, when chosen as above. After initialization, new training data was carefully chosen from the image to which ssFCM was applied. This training data was used by ssFCM in all further calculations replacing the \( U^d \) matrix used for initialization. The segmentation that \( k\)-nn provided using this data was considered very good. Hence, the training data may be considered good. The result of this experiment with training weights as low as \( w_i = 1 \) \( V^i \) is that ssFCM gave a correct segmentation and labeling, performed in 22 min 30 s on a SPARC2. With \( w_i = 6 \) \( V^i \), all clusters of interest are recognized and labeled and the resulting segmented image is closer to that obtained using \( k\)-nn; this segmentation takes 15 min 30 s. So with reliably good training data, ssFCM results in a faster and more accurate segmentation when more weight is given to the training data. We conclude that it is possible for good training data to overcome a bad initialization.

7. CONCLUSIONS

The ssFCM method allows the interjection of training data into an unsupervised clustering scheme. This overcomes the problems of determining the correct number of clusters and insuring that the algorithm provides correct physical labels, when (at least some) training data from every class is available. A judicious choice of weights for the training data provides clustering algorithms that use a sum of squared errors objective function a way to circumvent their tendency to produce partitions in which each class is of approximately equal size.

The location of the cluster prototypes or centers is influenced by the fixed training data. As the training data is given more weight, terminal cluster centers are more rigidly attracted to the centroids of the training data. Hence, the training data have an increasingly stronger influence on the resulting partition as higher weights are assigned to the training data for specific classes. A small amount of training data can be used to strongly influence the clustering process when it is considered accurately labeled. Alternatively, the method allows training data that may not be considered perfect to influence the final clusters to a desired degree. Note, it is not necessary to have training data from all classes in the ssFCM algorithm although it is assumed in this paper. Expert bias is injected into the clustering process in the choice of labeled training data and, more importantly, in the choice of the weight for the set of training patterns for each class.

The ssFCM algorithm has a distinct advantage over fully supervised algorithms when the training data is not known to be totally reliable. The final partition will only be influenced by the training data to a small degree, determined by low weights given to the training data (i.e. small \( w^i \)). Hence, poor training data will not necessarily ruin the final partition. Also, ssFCM may be effectively applied with less training data than other supervised methods.

The results of experiments on MR images and our small artificial data set are very promising. They show that the use of ssFCM avoids problems P1 and P2 and can overcome problem P3 from unsupervised clustering. The algorithm also is effective in providing good segmentations of MR images with similar results (as in Fig. 5) in other experiments that have been run on multiple slices of the same brain (to produce 3-D segmentations). The algorithm is the least sensitive algorithm to the choice of training data in our tumor volume measurement experiments. With an effective choice of parameters, ssFCM provides a good alternative classification algorithm that has the desirable characteristics of both fully supervised and unsupervised algorithms.

Acknowledgements—This research was partially supported by a grant from the Whitaker Foundation and the National Cancer Institute under grant CA59 425-01.

REFERENCES

Methods, Artificial Neural Networks in Engineering Conference, ANNIE (1994).


APPENDIX A: DATA FOR EXAMPLE 1

In (x, y) coordinate format.

Class 1

(24, 51), (25, 59), (27, 38), (28, 46), (29, 62), (30, 30), (33, 54), (34, 42), (34, 37), (35, 48), (36, 64), (39, 57), (39, 29), (40, 42), (40, 46), (41, 63), (41, 51), (42, 33), (39, 68), (43, 58), (45, 46), (47, 68), (48, 55), (48, 39), (48, 29), (49, 60), (50, 52), (52, 45), (53, 38), (53, 31), (55, 64), (56, 55), (58, 59), (58, 44), (60, 41), (60, 50), (61, 36), (63, 62), (64, 3), (65, 45).

Class 2

(89, 50), (90, 48), (94, 49).

About the Author—LAWRENCE O. HALL is an Associate Professor of Computer Science and Engineering at University of South Florida. He received his Ph.D. in computer science from the Florida State University in 1986 and a B.S. in applied mathematics from the Florida Institute of Technology in 1980. His current research in Artificial Intelligence is in hybrid connectionist, symbolic learning models, medical image understanding through integrated pattern recognition and AI, parallel expert systems, the use of Fuzzy sets and logic in imprecise domains and learning imprecise concepts such as tall or fast. He is an Associate Editor for IEEE Transactions on Systems, Man and Cybernetics, and IEEE Transactions on Fuzzy Systems. He is the President of the North American Fuzzy Information Processing Society. Dr Hall has written over 100 research papers and co-authored one book.

About the Author—AMINE M. BENSAID received the B.S. degree in information systems in 1990, the M.S. in computer engineering in 1992 and the Ph.D. in computer science and engineering in 1994, from the University of South Florida. Amine is currently an Assistant Professor of Computer Science at Al Akhawayn University, Ifrane, Morocco. His research interests include pattern recognition, image segmentation, fuzzy logic, neural networks and genetic algorithms. He is a member of IEEE, Tau Beta Pi and Phi Kappa Phi.

About the Author—JIM BEZDEK received his Ph.D. from Cornell University in 1973. His interests include pattern recognition, fishing, computational neural networks, skiing, image processing, blues music, medical computing and motorcycles. He is founding editor of the IEEE Trans. Fuzzy Systems and a fellow of the IEEE.

About the Author—LAURENCE P. CLARKE, Ph.D., is Professor of Radiology and Physics, College of Medicine, and Senior Member in Residence at the H. Lee Moffitt Cancer Center and Research Institute and Director of the Moffitt Digital Medical Imaging Program at the University of South Florida. Dr Clarke is a Fellow of the American Association of Physicists in Medicine and Society of Magnetic Resonance Imaging and member of the IEEE EMB and SNM Society. Research interests include computer assisted diagnosis (CAD) in medical imaging for all imaging modalities. Dr Clarke can be reached at the Department of Radiology, College of Medicine, University of South Florida, Tampa, FL 33612-4799, U.S.A.; fax (813) 975-7831; email: clarko@splinter.rad.usf.edu.