Fuzzy c-Shape: A new algorithm for clustering finite time series waveforms

Fateme Fahiman*, James C.Bezdek†, Sarah M.Erfani†, Marimuthu Palaniswami* and Christopher Leckie†
*Department of Electrical and Electronic Engineering, University of Melbourne, Australia
Email: ffahiman@student.unimelb.edu.au; palani@unimelb.edu.au
†Department of Computing and Information Systems, The University of Melbourne, Australia
Email: {jbezdek,sarah.erfani, caleckie}@unimelb.edu.au

***Abstract*—The existence of large volumes of time series data in many applications has motivated data miners to investigate specialized methods for mining time series data. Clustering is a popular data mining method due to its powerful exploratory nature and its usefulness as a preprocessing step for other data mining techniques. This article develops two novel clustering algorithms for time series data that are extensions of a crisp c-shapes algorithm. The two new algorithms are heuristic derivatives of fuzzy c-means (FCM). Fuzzy c-Shapes plus (FCS+) replaces the inner product norm in the FCM model with a shape-based distance function. Fuzzy c-Shapes double plus (FCS++) uses the shape-based distance, and also replaces the FCM cluster centers with shape-extracted prototypes. Numerical experiments on 48 real time series data sets show that the two new algorithms outperform state-of-the-art shape-based clustering algorithms in terms of accuracy and efficiency. Four external cluster validity indices (the Rand index, Adjusted Rand Index, Variation of Information, and Normalized Mutual Information) are used to match candidate partitions generated by each of the studied algorithms. All four indices agree that for these finite waveform data sets, FCS++ gives a small improvement over FCS+, and in turn, FCS+ is better than the original crisp c-shapes method. Finally, we apply two tests of statistical significance to the three algorithms. The Wilcoxon and Friedman statistics both rank the three algorithms in exactly the same way as the four cluster validity indices.**

I. Introduction

Increasing large volumes of time series data are becoming available from a diverse range of sources, including smart phones, environmental sensors and biological devices. Techniques for analyzing time series data include dimensionality reduction [1], indexing [2], [3], and segmentation [4]. In addition, techniques have been developed to extract the underlying shape or trends in the data, for tasks such as pattern discovery [5]–[7], classification [8], [9], rule discovery and summarization. An open challenge in time series analysis is how to cluster a set of time series according to the similarity of their underlying shape. The basic objective of this article is to generalize the fuzzy c-means (FCM) model [10] to accommodate clustering in finite time series data.

Most state-of-the-art approaches for shape-based clustering suffer from two main drawbacks: *(i)* they are computationally expensive and thus, not suitable for large volumes of data [6], [7], [11], [12]; and *(ii)* these approaches are limited to specific domains [12], or their effectiveness has only been evaluated over a small number of datasets [7], [11].

k-Shape [13] is a novel method that has been proposed to address the problem of finding a suitable distance measure and clustering method for finite time series data, where time series sequences belong to the same cluster, exhibiting similar patterns, regardless of differences in amplitude and phase. To demonstrate the robustness of k-Shape clustering [13], the authors perform an extensive evaluation over 48 different time series datasets and demonstrate the superiority of their technique against fifteen existing schemes.

In this paper, we introduce two novel clustering algorithms. The first substitutes the *Shape Based Distance* (SBD) measure introduced in [13] for the model norm in the standard fuzzy c-means [14] algorithm, and the second uses the SBD measure and shape extraction method in [13] to update prototypes in the fuzzy c-means clustering algorithm [14]. These two modifications result in 4.5% and 58.5% improvement respectively in accuracy and efficiency in comparison to k-Shape clustering. We make a direct comparison with the results reported in the k-Shape paper, by evaluating our methods on the same UCR time series classification archive [15], which consists of 48 labeled time series datasets from various real-world domains.

The rest of the paper is organized as follows: Section II presents the relevant theoretical background on fuzzy c-means and k-Shape clustering that are based of our two new algorithms. Section III introduces our two novel clustering methods FCS++ and FCS+. Section IV discusses the four external cluster validity indices used for measuring the accuracy of clustering methods. Section V evaluates our proposed methods with extensive numerical experiments, and shows that our algorithms outperform the crisp k-Shape clustering algorithm in terms of accuracy and efficiency. Section VI contains our conclusions and some future research directions.

II. Preliminaries

In this section we review the relevant theoretical background for our new clustering algorithms. In Section II-A, we review fuzzy c-means clustering, which is the basis of our work. In Section II-B, we introduce the shape-based distance (SBD) [13]. Section II-C introduces a new method for assigning a center to clusters based on the shape of the time series and briefly explains the k-Shape clustering method [13].

978-1-5090-6034-4/17/$31.00

A. The Generalized Fuzzy c-Means Model

Crisp and fuzzy/probabilistic c-partitions of n objects are, respectively, the sets of matrices defined as

$$M_{fcn} = \{U \in \Re^{cn} : for\, 1 \le i \le c, 1 \le k \le n; 0 \le u_{ik} \le 1; \sum_{i=1}^{c} u_{ik} = 1\ \forall k; \sum_{k=1}^{n} u_{ik} > 0\ \forall i\} \quad (1a)$$

$$M_{hcn} = \{U \in M_{fcn} : u_{ik} \in \{0,1\} \forall i,k\}. \quad (1b)$$

The *Generalized Least Squares Error* (GLSE) clustering model is the constrained optimization problem

$$min_{(U,P)}\{J_m(U,P;X) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m[\Delta(x_k,p_i)]^2\}. \quad (2)$$

In equation (2), m is a weighting parameter, $m \ge 1$, $U \in M_{fcn}$, $P = \{p_1, \ldots, p_c\}$ is a set of cluster prototypes, and $[\Delta(x_k, p_i)]^2$ is a measure of the squared error incurred by representing input x_k by prototype p_i. Optimal partitions U^* of X are taken from pairs (U^*, V^*) that are local minimizers of J_m.

There are dozens of instances of equation (2) for specific choices of P and Δ. For example, the prototypes P may be points, lines, planes, linear varieties [10], hyperquadrics [16], or even regression functions [17]. The dissimilarity measure Δ may be an inner product norm [10], a Minkowski norm [18], or a shape-based distance [13] such as the one used in this paper.

Theorem (1) (FCM) [14]: Let $X = \{x_1, \ldots, x_n\} \subset \Re^p$ contain at least c distinct points. Let $P = V = \{v_1, \ldots, v_c\} \subset \Re^p$, and A be a positive-definite norm inducing $p \times p$ weight matrix, $\Delta(x_k, p_i)^2 = \|x_k - v_i\|_A^2 = (x_k - v_i)^T A(x_k - v_i) > 0 \quad \forall i,k$. If $m > 1$, then $(U,V) \in M_{fcn} \times \Re^p$ may minimize $J_m(U,V;X)$ only if

$$u_{ik} = \left(\sum_{j=1}^{c}\left(\frac{\|x_k - v_i\|_A}{\|x_k - v_j\|_A}\right)^{\frac{2}{m-1}}\right)^{-1}; 1 \le i \le c; 1 \le k \le n, \quad (3a)$$

$$v_i = \sum_{k=1}^{n} u_{ik}^m x_k / \sum_{k=1}^{n} u_{ik}^m;\ 1 \le i \le c. \quad (3b)$$

It is also well known that when $m = 1$ the partition matrix U is necessarily crisp, $U \in M_{hcn}$, and under the remaining hypotheses in Theorem (1), that equation (2) reduces to the classical hard c-means (HCM, or classical k-means) model which was first discussed by Lloyd in [19].

Theorem (2) (HCM) [14] : Let $X = \{x_1, \ldots, x_n\} \subset \Re^p$ contain at least c distinct points, $P = V = \{v_1, \ldots, v_c\} \subset \Re^{cp}$, and $m = 1$. If $\Delta(x_k, p_i)^2 = \|x_k - v_i\|_A^2 > 0 \quad \forall i,k$, then $(U,V) \in M_{fcn} \times \Re^{cp}$ may minimize $J_1(U,V;X)$ only if $U \in M_{hcn}$ is a *hard c-partition* of X, and

$$u_{ik} = \begin{Bmatrix} 1 & \|x_k - v_i\|_A < \|x_k - v_j\|_A \forall j \ne i \\ 0 & \text{otherwise} \end{Bmatrix} \quad (4a)$$

$$v_i = \sum_{k=1}^{n} u_{ik} x_k / \sum_{k=1}^{n} u_{ik};\ 1 \le i \le c \quad (4b)$$

The content of Theorem (2) is often given by denoting the partition $U \in M_{hcn}$ by its equivalent set-theoretic form, $X = \bigcup_{i=1}^{c} X_i; \oslash = X_i \bigcap_{i \ne j} X_j; \sum_{i=1}^{c} \mid X_i \mid = \sum_{i=1}^{c} n_i = n$. Using this representation, equations (4) take the alternate form

$$u_{ik} = \begin{Bmatrix} 1 & x_k \in X_i \\ 0 & \text{otherwise} \end{Bmatrix}; 1 \le i \le c,\ 1 \le k \le n \quad (5a)$$

$$v_i = \bar{v}_i = \sum_{x_k \in X_i}\left(\frac{x_k}{n_i}\right); 1 \le i \le c. \quad (5b)$$

The advantage of using equations (4) instead of (5) is that the role of the partition matrix U is clearly seen in the more general fuzzy case. Note that (5a) labels each point in X with the nearest prototype rule; and (5b) shows that the point prototypes are none other than the geometric centroids (sample means) of the c clusters in X.

There are various ways to estimate solutions for the GLSE problem (2) for the choice $[\Delta(x_k, p_i)]^2 = \|x_k - v_i\|_A^2$. The most popular method is Picard iteration through necessary conditions (3) or (4). For the fuzzy case, these are *first order necessary conditions* (FONCs, the gradient must vanish at extreme points). For the crisp case, looping through conditions (4) or (5) is sometimes called Lloyd iteration: (4b), (5b) is a FONC, and (4a), (5a) is necessary, but not first order. This method is summarized as Algorithm 1.

Algorithm 1 AO c-means for HCM and FCM

Input: $X = \{x_1, \ldots, x_n\} \subset \Re^p$
Output: (U,V)
1: Pick $1 < c < n : m \ge 1$ $T_M = iterationLimit$
2: Model Norm: $\| x - v \|_A = \sqrt{(x-v)^T A(x-v)}$
3: $E_t = \mid J_m(U_t, V_t) - J_m(U_{t-1}, V_{t-1}) \mid$
4: $0 < \varepsilon = terminationCriteria$
5: Guess $V_0 = \{v_{10}, \ldots, v_{c0}\} \subset \Re^{cp}$
6: Calculate U_0 with V_0 and (3a) or (4a)
7: $t = 1$: $E_1 = bigNumber$
8: **while** $t < T_m \quad and \quad E_t > \varepsilon$ **do**
9: Calculate U_t with V_{t-1} and (3a) or (4a)
10: Calculate V_t with (3b) or (4b)
11: $t = t + 1$
12: **end while**
13: $(U,V) \leftarrow (U_t, V_t)$

Our interest is confined to three algorithms for clustering finite time series in the special case when each $x_k \in X$ is a vector of length p that represents a finite portion of a waveform or time series. All three algorithms are ad hoc methods that

depend on equations (4) or (5) and use Algorithm 1 with appropriate modifications to find approximate solutions.

B. Shape Based Distance

The quality of clustering in feature vector data almost always depends on finding a good way to measure similarity or distance between the items represented by the data. Most of the extant literature on clustering in time series data modifies a classic clustering algorithm such as k-means by *(i)* substituting a suitable distance measure on the raw data; or *(ii)* by extracting features from the data that convert it into vectors that can be inserted directly into a classic algorithm. This paper follows the approach taken in [13], which concentrates attention on SBD as the distance of choice for waveform clustering.

The new algorithms begin by modifying the GLSE model at (2) by introducing the SBD, which is tailored to the time series case to define $\Delta(x_k, p_i)$. Specifically, this distance is introduced for a pair of z-normalized vectors x and y of length p in Algorithm 2:

Algorithm 2 [13]: $[dist, y'] = SBD(x, y)$

Input: Two z-normalized sequences $x, y \in \Re^p$
Output: Dissimilarity dist = SBD(x,y) $\in \Re^+$;Aligned sequence y' of y towards x

1: length = $2^{nextpower2(2*length(x)-1)}$
2: $CC = IFFT\{FFT(x, length)^* FFT(y, length)\}$%(A)
3: $NCC_c = CC/ \parallel x \parallel . \parallel y \parallel$
4: [value,index] = max(NCC_c)
5: dist= 1 - value % (B)
6: shift = index - length(x)
7: **if** $shift \geq 0$ **then**
8: y'= [zeros(1,shift), y(1:end-shift)] %(C)
9: **else**
10: y'= [y(1-shift:end), zeros(1,-shift)] %(C)
11: **end if**

Notes about Algorithm 2 The z-normalization mentioned in the input line is the standard $(0, 1)$ statistical normalization. Lines 2, 5, and 8 correspond to the following equations from [13].

(A) $CC(X, Y) = \mathcal{F}^{-1}\{\mathcal{F}(x)^* \mathcal{F}(y)\}$, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are forward (inverse) discrete Fourier transforms, $*$ is convolution.

(B) $dist = SBD(x, y) = 1 - max_w\{CC_w(x, y)/\|x\|.\|y\|\}$, where $w \in \{1, 2, \ldots, 2p-1\}$; $CC_w(x, y) = R_{w-p}(x, y)$; and

$$R_k(x, y) = \left\{ \begin{array}{ll} \sum_{j=1}^{p-k} x_{j+k} y_j; & k \geq 0 \\ R_{-k}(x, y); & k < 0 \end{array} \right\}.$$

$$(C)\ x_{(s)} = \left\{ \begin{array}{ll} (\overbrace{0, \ldots, 0}^{|s|}, x_1, \ldots, x_{p-s}); & s \geq 0 \\ (x_{1-s}, \ldots, x_p, \underbrace{0, \ldots, 0}_{|s|}); & s < 0 \end{array} \right\}.$$

Equation (C) shows how a sliding window passes across a vector, while looking for the optimal alignment between x and y. The inputs to Algorithm 2 are vectors in $\Re^p$, so the distance $\|x_k - v_i\|_A$ in equations (4a) and (5a) can be directly replaced by $SBD(x, y)$, which is called a distance in [13].

Paparrizos and Gravano argue in [13] that most of the research on waveform clustering has used distances like DTW to the exclusion of *cross correlation* (CC) between time series because this time honored statistical method suffers from normalization and registration issues. They assert that to be useful, the data and the CC applied to it must be appropriately normalized. The z-normalization of the inputs to Algorithm 2 gives it scale invariance, and takes care of inherent distortion in the data. They tackle the normalization of CC with equations (A)-(C) that follow Algorithm 2. Shift invariance is addressed by computing $CC_w(x, y) = R_{w-p}(x, y)$, which maximizes CC at the best match between x and y by testing each position offered by the sliding window in equation (C). They discuss three ways to normalize CC, but only use CC_w, shown and used here, which produces values in the closed interval $[-1, 1]$. Consequently, $SBD(x, y) \in [0, 2]$, and it takes the value 0 when x and y are perfectly similar (corresponding to zero distance, i.e., $x = y$). Efficiency of finding CC_w is addressed by using the discrete forward and inverse Fourier transforms and padding the input data so its length is always a power of 2. The overall complexity of Algorithm 2 is given as $O(plog(p))$.

C. Shape Based Prototypes and k-Shape Clustering

The second major alteration of k-means introduced in [13] is to compute prototypes that attempt to capture shape information, based on the fact that input vectors represent finite time series waveforms. To begin, we revisit equation (3a) for v_i. This prototype arises for FCM by zeroing the gradient of the function $J_m(U^*, V; X)$ in the reduced optimization problem

$$min_{V \in \Re^{cp}} \left\{ J_m(U^*, V; X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik}^*)^m \|x_k - v_i\|_A^2 \right\}. \quad (6)$$

U^* is fixed in (6), and minimization of $J_m(U^*, V; X)$ is unconstrained, so solving $\nabla_{v_i} J_m(U^*, V; X) = 0$ for v_i leads directly to (3a) in the fuzzy case, and (4a) in the crisp case. When the distance in equation (6) is not an inner product norm, $J_m(U^*, V; X)$ is generally not differentiable with respect to v_i. In this more general case, the reduced problem for the crisp case ($m = 1$ in (6)) of the GLSE model becomes:

$$min_{v \in \Re^p} \left\{ J_1(v; X) = \sum_{x_k \in X_i} \Delta(x_k - v)^2 \right\}. \quad (7)$$

The method of solving (7), sometimes referred to as the Steiner sequence problem [20], depends on the nature of the distance function $\Delta(x_k - v)$.Bypassing some intermediate steps given

in [13], (7) eventually becomes an instance of maximization of the famous Rayleigh Quotient [21], viz,

$$v_i = max_{v \in \Re^p} \left\{ \frac{v^T Q^T S Q V}{v^T v} \right\} = max_{v \in \Re^p} \left\{ \frac{v^T M v}{v^T v} \right\} \quad (8)$$

where $M = Q^T S Q$, $Q = I - (1/p)[\mathbf{1}]$, and $[\mathbf{1}]$ is the $p \times p$ matrix of $1's$. The solution of (8) is the eigenvector corresponding to the largest eigenvalue of the real symmetric matrix M, which is computed in the last line of Algorithm 3, which records this procedure as shown in [13].

Algorithm 3 [13]: $V_{i,t+1}$= **Shape Extraction**$(X, v_{i,t})$

Input: $X = \{x_1, \ldots, x_n\} \subset \Re^p$ as a $n \times p$ matrix X whose columns are z-normalized time series vectors. $v_{i,t} \in \Re^p$ is the reference sequence against which time series of X are aligned

Output: $v_{i,t} \in \Re^p$ % new i^{th} shape prototype

1: $X' \leftarrow [\quad]$
2: **for** $i = 1$ **to** n **do**
3: [dist,x'] $\leftarrow$ SBD$(v_{i,t}, X(i))$ %Algorithm 2
4: $X' \leftarrow [X'; x']$
5: **end for**
6: $S \leftarrow X'^T \bullet X'$ %S, Eq (8)
7: $Q \leftarrow I - \frac{1}{p}[\mathbf{1}]$ %Q, Eq (8)
8: $M \leftarrow Q^T \bullet S \bullet Q$ %M, Eq (8)
9: $v_{i,t} \leftarrow Eig(M, 1)$ %Extract 1^{st}ev

Now everything is in place for Papparrizos and Gravano to define their k-Shape method, repeated here from [13].

III. FCS+ AND FCS++ CLUSTERING

The SBD algorithm, in conjunction with Theorem (1), offers a way to immediately modify the basic FCM algorithm for waveform inputs. We can replace the inner product norm in Line 2 of Algorithm 1 with the SBD computed by Algorithm 2. This results in our first heuristic generalization, which we will call *fuzzy c-Shape plus* (FCS+). We have added the "+" to the acronym FCS so that you won't confuse this new shape-based distance algorithm with an older one bearing the acronym FCS (*fuzzy c-shells*, [22]).

Note about Algorithm 5. The optional lines of Algorithm 5 harden the terminal fuzzy partition. This option is NOT necessary, since there are soft versions of all four CVIs based on the contingency matrix discussed in the next section [23]. However, we will use this option for FCS+ in our numerical experiments so that the comparison of FCS+ to k-Shape (which has only crisp partitioning) is equitable. Algorithm 4 represents the crisp partition of X that it produces as the $n \times 1$ label vector **u** (not bold in Algorithm 4). This is an efficient way to carry the information in the crisp case. To see how a fuzzy generalization would make sense, let us represent the partition information possessed by the crisp vector **u** as a matrix $U \in M_{hcn}$. For example, suppose the output of Algorithm 4 is $\mathbf{u} = [12131]$, so there are $c = 3$ crisp

Algorithm 4 [13]: $[u, V]$= **k-Shape**(X, c)

Input: $X = \{x_1, \ldots, x_n\} \subset \Re^p$ as a $n \times p$ matrix X whose columns are z-normalized time series vectors: c is the number of clusters produced

Output: **u** is a $n \times 1$ label vector that partitions X into c crisp clusters. V is a $c \times p$ matrix containing c shape extracted centroids of length p.

1: $iter \leftarrow 0$
2: $u' \leftarrow [\quad]$
3: **while** $u! = u'$ and $iter < 100$ **do**
4: $u' \leftarrow u$
% Refinement step
5: **for** $j \leftarrow 1$ to c **do**
6: $X' \leftarrow [\quad]$
7: **for** $i \leftarrow 1$ to n **do**
8: **if** u(i)= j **then**
9: $X' \leftarrow [X', X(i)]$
10: **end if**
11: **end for**
12: $v(j) \leftarrow$ Shape Extraction $(X'; v(j))$ %Algorithm3
13: **end for**
% Assignment step
14: **for** $i \leftarrow 1$ to n **do**
15: $mindist \leftarrow \infty$
16: **for** $j \leftarrow 1$ to c **do**
17: $[dist, x'] \leftarrow$ SBD $(v(j), X(i))$ %Algorithm2
18: **if** $dist < mindist$ **then**
19: $mindist \leftarrow dist$
20: $u(i) \leftarrow j$
21: **end if**
22: **end for**
23: **end for**
24: $iter \leftarrow iter + 1$
25: **end while**

Algorithm 5 FCS+

1: **Replace**: line 3 in Algorithm FCM with: 3 Model Norm: SBD$(x, v) \leftarrow \| x - v \|_A^2$
2: **Do**: Algorithm 1
3: $[optional]$ Harden U, line 13, Algorithm 1 with Eq(11)
4: $[optional]$ Output: $H_{mm}(U) \in M_{hcn}$

labels for $n = 5$ objects. The matrix representation for this **u** is

$$U = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (9)$$

During the refinement step, Algorithm 4 will update the three cluster centers by calling Algorithm 3 three times. Each prototype update uses only the data vectors in its cluster, illustrated graphically as follows:

$$U = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} \Longrightarrow & v_{1,t} & \rightarrow & v_{1,t+1} \\ \Longrightarrow & v_{2,t} & \rightarrow & v_{2,t+1} \\ \Longrightarrow & v_{3,t} & \rightarrow & v_{3,t+1} \end{matrix} \quad (10)$$

Equation (10) shows how *crisp* membership in each cluster is used to control which vectors in the data set are accessed during the update procedure. At each j, lines 6-9 of Algorithm 4 picks out only the vectors in current cluster X_j (corresponding to the 1's in the j^{th} row of U) and writes them into $X^{'}$, so when Algorithm 3 is called in line 12 of Algorithm 4, only the points in X_j are sent to it via array $X^{'}$ to update the shape based prototype for the j^{th} cluster.

The representation of the partition produced by Algorithm 4 at (10) shows how to generalize the k-Shape algorithm to the fuzzy case. Algorithm FCS+ produces a fuzzy $U \in M_{fcn}$, which we can harden using the maximum membership function $\mathbf{H}_{mm} : M_{fcn} \rightarrow M_{hcn}$, which operates on the columns of U. $\mathbf{H}_{mm}(U) = [\mathbf{h}(\mathbf{U}^{(1)}) \cdots \mathbf{h}(\mathbf{U}^{(n)})]$ is a hardening of U defined on the n columns $\{\mathbf{U}^{(k)}\}$ of U as follows:

$$\mathbf{h}(\mathbf{U}^{(k)}) = (0, 0, \ldots, \underbrace{1}_{i^{th}}, \ldots)^T$$

$$\Longleftrightarrow \begin{Bmatrix} 1 & u_{ik} > u_{jk}; j \neq i \\ 0 & \text{otherwise} \end{Bmatrix}. \quad (11)$$

In words: **h** replaces the largest value in each column of U with a 1, and place 0's in the other $c-1$ slots in each column of U. When ties occur, assign the membership 1 arbitrarily to any winner, and treat the other maximums as non-winners. For example if we apply H_{mm} to the matrix

$$U' = \begin{bmatrix} 0.9 & 0.1 & 0.6 & 0.3 & 0.65 \\ 0.1 & 0.8 & 0.3 & 0.22 & 0 \\ 0 & 0.1 & 0.1 & 0.75 & 0.35 \end{bmatrix} \quad (12)$$

the result is that $\mathbf{H}_{mm}(U^{'}) = U$, the matrix at (10).

Once this conversion is made, it is a simple matter to convert $\mathbf{H}_{mm}(U^{'})$ back to the list form row by row, $\mathbf{u} \leftarrow \mathbf{H}_{mm}(U^{'})$, required as one of the inputs to Algorithm 4. With this conversion in hand, we can define the *fuzzy c-Shape* double plus (FCS++) algorithm, where the first + stands for the use of the SBD distance function (Algorithm 2), and the second + represents the use of SE prototypes via Algorithms 3 and 4.

IV. Cluster Validity Indices

The quality of our experimental outputs can be judged in a number of ways. To make direct comparisons between the k-Shape outputs in [13] to those found by FCS+ and FCS++, we will follow [13] by using external *cluster validity indices* (CVIs), which are functions that identify a "best" member amongst a set of *candidate partitions* $CP = \{U \in M_{fcn} : c_m \leq c \leq c_M\}$ of any set of $O = \{o_1, \ldots, o_n\}$ of n objects.

Chapter 16 in [21] is an excellent source of general information about CVIs.Here we briefly summarize the four external

Algorithm 6 FCS++

Input: $X = \{x_1, \ldots, x_n\} \subset \Re^p$
Output: (U, V)
1: **Pick** $1 < c < n : m \geq 1 : T_M$= iteration limit
2: Model Norm: $\Delta(x, v) = \text{SBD}(x, v)$
3: Error Norm: $E_t = \parallel V_t - V_{t-1} \parallel$
4: $0 < \varepsilon =$ termination criterion
5: **Guess** $V_0 = \{v_{10}, \ldots, v_{c0}\} \subset \Re^{cp}$
6: $t = 1 : E_1 =$ big number
7: **while** $T < T_M$ and $E_t > \varepsilon$ **do**
8: Calculate U_t with V_{t-1} and (3a)
9: $\mathbf{u} \leftarrow \mathbf{H}(u_t)$ %Harden U_t with Eq(11)
%SE Refinement step
10: **for** $j \leftarrow 1$ to c **do**
11: $X' \leftarrow [\ \ ]$
12: **for** $i \leftarrow 1$ to n **do**
13: **if** $u(i) = j$ **then**
14: $X' \leftarrow [X', X(i)]$
15: **end if**
16: $v(j) \leftarrow$ Shape Extraction $(X'; v(j))$ %Algorithm3
17: **end for**
18: **end for**
19: $V_t = [V(1)V(2) \cdots v(c)]$
20: $t = t + 1$
21: **end while**
22: $(U, V) \leftarrow (u_t, V_t)$ $\quad$ % $U \in M_{hcn}$ is crisp

validity indices we use to evaluate our algorithms.

Let $U \in M_{hrn}$, $Q \in M_{hcn}$ and $N = UQ^T$ (in general, $r \neq c$). When a reference (ground truth) partition is available, it will be Q in the transformation $N = UQ^T$. The matrix N forms a contingency table between the two partitions, so is sometimes called a *contingency matrix*. Given Q, we match Q to candidate $U \in CP$ using $CVI(U \mid Q)$. The Rand index [24], one of the first (and still most popular) crisp external CVIs, is based on (4) paired comparison values derived from the elements of N :

$$a = \frac{1}{2} \sum_{i=1}^{r} \sum_{j=1}^{c} n_{ij}(n_{ij} - 1); \quad (13a)$$

$$b = \frac{1}{2} \left(\sum_{j=1}^{c} n_{\bullet j}^2 - \sum_{i=1}^{r} \sum_{j=1}^{c} n_{ij}^2 \right); \quad (13b)$$

$$c = \frac{1}{2} \left(\sum_{i=1}^{r} n_{i\bullet}^2 - \sum_{i=1}^{r} \sum_{j=1}^{c} n_{ij}^2 \right); \quad (13c)$$

$$d = \frac{1}{2} \left(n^2 + \sum_{i=1}^{r} \sum_{j=1}^{c} n_{ij}^2 - (\sum_{i=1}^{r} n_{i\bullet}^2 + \sum_{j=1}^{c} n_{\bullet j}^2) \right). \quad (13d)$$

CVIs are notoriously fickle, so we use four CVIs here based on different combinations of the values in the contingency matrices to see if they will all rank the three algorithms the

same way. The *Rand Index* (RI) is computed with these four values as

$$RI(U \mid Q) = \frac{(a+d)}{(a+d)+(b+c)} \tag{14}$$

The numerator $(a+d)$ is the number of $agreements$ between pairs in U and Q; $(b+c)$ is the number of $disagreements$. The RI is valued in $[0,1]$, taking its maximum if and only if $U = Q$. So, the heuristic for this index is that the maximum value of the RI over the partitions in CP points to the best match among the candidates. We call such an index a $max - optimal \quad CVI$, indicated as $(\uparrow)$. Several adjustments of (14) have been proposed in the literature that attempt to rectify the tendency of the RI to increase monotonically with c. Of these, the *Adjusted Rand Index* (ARI) of Hubert and Arabie [25] is the most popular:

$$ARI(U \mid Q) = \frac{\left(a - \frac{(a+c)(a+b)}{a+b+c+d}\right)}{\left(\frac{(a+c)+(a+b)}{2} - \frac{(a+c)(a+b)}{a+b+c+d}\right)}. \tag{15}$$

The ARI is also max-optimal $(\uparrow)$: it maximizes at 1, but its minimum may be negative if the index is less than its expected value of zero. The third external CVI we will use is the *variation of information* (VI) introduced in [26]:

$$VI(U \mid Q) = -\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{n_{ij}}{n} log \frac{n_{ij}}{n} - \left[\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{n_{ij}}{n} log \frac{n_{ij}/n}{(n_{i\bullet}/n)(n_{\bullet j}/n)}\right]. \tag{16}$$

In equation (16) $n_{i\bullet}$, $n_{\bullet j}$ are, respectively, the i^{th} row and j^{th} column sums of the contingency matrix N.

The VI index is a metric valued in $[0, log(n)]$ which takes the value 0 when $U = Q$, so the heuristic for VI is to accept the partition achieving the minimum value over CP: the VI is min-optimal $(\downarrow)$.

The last external CVI we use is a form of mutual information, which is normalized by a maximum calculation, so this index bears the notation $NMI_M(U \mid Q)$:

$$NMI_M(U \mid Q) = \frac{\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{n_{ij}}{n} log \frac{n_{ij}/n}{(n_{i\bullet}/n)(n_{\bullet j}/n)}}{max\{H_S(U), H_S(Q)\}}, \tag{17}$$

where, for example, $n_i = \sum_{k=1}^{n} u_{ik}$ is the number of points in the i^{th} cluster in U, and $H_S(U) = -\sum_{k=1}^{r}\sum_{i=1}^{c}\frac{n_i}{n} log \frac{n_i}{n}$ is Shannon's entropy of U [21], and likewise for $H_S(Q)$.The range of $NMI_M(U \mid Q)$ is $[0,1]$, and it is a max-optimal $(\uparrow)$ CVI.

V. Numerical Experiments

We use the same 48 synthetic and real data sets that were used in [13]. The 48 sets are all z-normalized, crisply labeled, split into training and test sets, and are collected in the UCR time series database [15].We ran the experiments for each clustering method ten times for each data set. We implemented the clustering methods in MATLAB R2014b (64bits) using the platform: Intel(R) with core(TM)i7 processor and clock speed at 3.60 GHz and 16 GB RAM.

Table I shows the grand averages over 480 trials for each of the four crisp external CVIs. Each index was evaluated 10 times for different runs of FCS+ and FCS++ on each of the 48 labeled data sets. The ordering for the three algorithms is the same for all four indices. Table I shows that FCS++ is slightly better than FCS+, and in turn, FCS+ outperforms k-Shape by a small margin for all four indices. This demonstrates that the overall quality of both of the new fuzzy methods for clustering waveform data is superior to the k-Shape algorithm in [13].

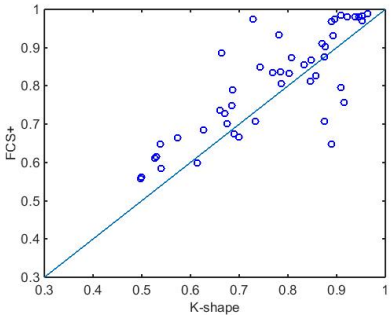
TABLE I. Grand Average of (480) CVI values
(10 runs per data set x 48 data sets)

CVI	Type	Range	FCS++	FCS+	k-Shape
$\overline{\overline{RI}}$	$(\uparrow)$	$[0,1]$	**0.822**	0.807	0.772
$\overline{\overline{ARI}}$	$(\uparrow)$	$[-a,1]$	**0.461**	0.403	0.321
$\overline{\overline{NMI_M}}$	$(\uparrow)$	$[0,1]$	**0.641**	0.534	0.413
$\overline{\overline{VI}}$	$(\downarrow)$	$[0, logn]$	**1.010**	1.463	2.455

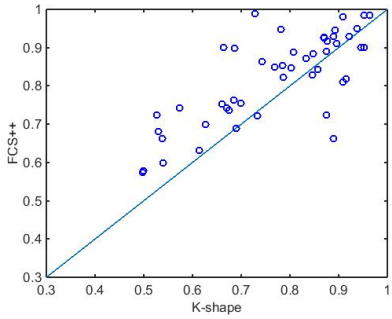
Figure 1 compares each pair of methods graphically. Each of the views 1(a), 1(b) and 1(c) plots the average values of the Rand Index over 10 runs for each of the 48 data sets. The dots represent the 48 data sets, and *the coordinates of each dot are the average values of the Rand index for the algorithm pair in each view*.

For example, the coordinates of points in Figure 1(a) are: horizontal coordinate x = average RI value achieved by 10 runs of k-Shape; vertical coordinate y = average RI value achieved by 10 runs of FCS+. There are 10 points below the line y = x in Figure 1(a), 1 point on the line, and 37 points above the line. This means that FCS+ achieved a better average result with the RI than k-Shape on 37 of the 48 data sets, they were tied on one data set, and k-Shape had a better average RI than FCS+ on 10 data sets. Figure 1(b) shows that FCS++ is slightly better, with 38 data sets above the line. And Figure 1(c) shows that FCS++ achieves a better result than FCS+ on 38 of the 48 data sets.

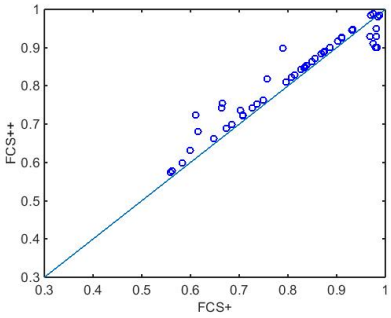
Time complexity analysis: Assume that n and p are the number and the length of the finite time series respectively and c is the number of clusters. All three algorithms use the SBD function to measure dissimilarity between data points and centroids. The SBD function requires $O(plog(p))$ time to calculate this measurement. The implementation of FCM as shown in Algorithm 1 is $O(npc^2)$. k-Shape uses the k-means algorithm as its



((a)) FCS+ vs. k-Shape. Circles above the diagonal indicate datasets for which FCS+ has a better average Rand Index than k-Shape.



((b)) FCS++ vs. k-Shape. Circles above the diagonal indicate datasets for which FCS++ has a better average Rand Index than k-Shape.



((c)) FCS++ vs. FCS+. Circles above the diagonal indicate datasets for which FCS++ has better average Rand Index than FCS+.

Fig. 1. Comparisons of k-Shape, FCS+ and FCS++ : average RI values of ten runs over 48 data sets.

TABLE II. Average run time for 480 values of the Rand Index (48 data sets, 10 runs each)

Algorithm	Ave. CPU Time (in seconds)
FCS+	9.51
k-Shape	22.95
FCS++	25.04

underlying clustering algorithm, where the time complexity of k-means is $O(npc)$. Now, the time complexity for the FCS+ algorithm can be calculated as $O(ncplog(p))$ time. In FCS++ and k-Shape clustering algorithms, there is a refinement step, which for every cluster calculates matrix M with $O(p^2)$ time complexity, and then computes an eigenvalue decomposition on M with $O(p^3)$ time complexity. Therefore, the complexity of the refinement step is $O(max\{np^2, cp^3\})$. As a result, the per iteration time complexity of FCS++ and k-Shape are $O(max\{nc^2plog(p), np^2, cp^3\})$ and $O(max\{ncplog(p), np^2, cp^3\})$ time respectively. Thus, all three algorithms are linear in the number of time series, and the major portion of the computational cost rests with p, the length of the time series.

Table II shows the average CPU time taken by each of the three algorithms to evaluate the Rand index for 10 runs of FCM on each of the 48 data sets. FCS+ takes roughly 9.5 secs per run, whereas the other two required about three times that, FCS++ topping out at about 25 secs per run. This is easy to understand: FCS+ does not use SE Algorithm 3, so it is considerably less expensive computationally than the other two algorithms. Overall, these algorithms are reasonable fast for the 48 data sets used in our experiments.

Statistical analysis: We used two statistical tests that assess the statistical significance of the performance of the various methods. First, we perform pairwise comparisons between different methods using the Wilcoxon signed-rank test [27]. The test returns a ρ-value associated with each comparison, representing the lowest level of significance of a hypothesis that results in a rejection. This value can be used to determine whether two algorithms have significantly different performance and to what extent. For all the comparisons in this study the significance level α is set to 0.05. Table III summarizes these comparisons on all the accuracy results from the three algorithms over 48 datasets. In the three mentioned tables, the ρ-value is less than the significance level ($\alpha = 0.05$), which means that there is a significant difference between the accuracy of algorithms. To illustrate, consider the *RI Accuracy*, where the first row compares k-Shape with FCS++. The sum of ranks for k-Shape is less than the sum of ranks for FCS++. Therefore, the accuracy of FCS++ is better than k-Shape clustering. Another statistical test that enables us to compare the three clustering algorithms is Friedman's test [28]. The Friedman test is a non-parametric statistical test for differences between groups. Figure 2 shows the results of applying Friedman's test to the three clustering algorithms with respect to the three max optimal CVIs used in our study. Friedman's test supports the conclusions drawn from Wilcoxon's test, viz., that FCS++ is superior to FCS+, which is in turn superior to k-Shape.

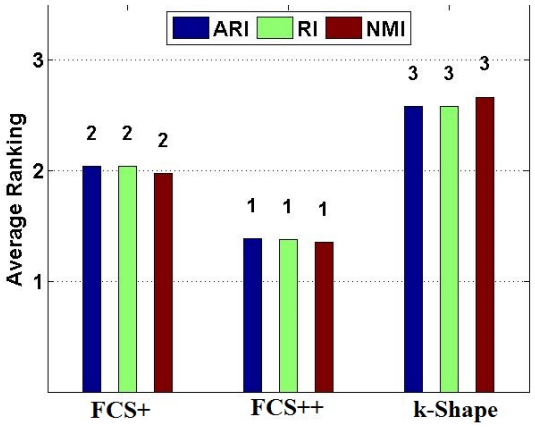


Fig. 2. Comparison for rankings of clustering accuracy methods for 3 metrics. The bars represent average rankings based on the Friedman test, and the number on the top of the bars indicates the ranking of the algorithm, from the best (1) to the worst (3) for each given measure. The ranking is determined for all datasets and finally an average is calculated as the mean of all rankings.

TABLE III. Wilcoxon test to compare three clustering methods in terms of accuracy based on RI, ARI, and NMI_M validity method. R^+ corresponds to the sum of the ranks for the method on the left and R^- for the right

	RI Accuracy			ARI Accuracy			NMI_M Accuracy		
Method	R^+	R^-	ρ-value	R^+	R^-	ρ-value	R^+	R^-	ρ-value
k-Shape vs FCS++	215	**913**	$1.304e^{-04}$	66.5	**1061.5**	$1.403e^{-07}$	75	**1053**	$1.426e^{-07}$
k-Shape vs FCS+	238	**813**	$3.309e^{-04}$	154	**897**	$6.848e^{-06}$	81	**970**	$1.991e^{-07}$
FCS++ vs FCS+	**931**	245	$2.608e^{-04}$	**1036.5**	139.5	$4.199e^{-06}$	**1009**	167	$4.199e^{-05}$

VI. CONCLUSIONS AND FUTURE REASERCH

Two new fuzzy variants of k-Shape were defined. FCS+ arises by substituting $SBD(x_k, p_i)$ for $\| x - v \|_A^2$ in FCM. FCS++ is FCS+ with the additional substitution of SE V for the prototypes in FCM. For FSC++, partitioning is adapted to the k-Shape algorithm by hardening the fuzzy partitions produced by FCM at each assignment step. We performed the same experiments with 48 labeled waveform data sets that were done using k-Shape in [13], and compared the two fuzzy methods to k-Shape using four crisp external cluster validity indices and two statistical tests of significance. All four indices indicate that FCS++ performs somewhat better than FCS+, and in turn, FCS+ is slightly superior to k-Shape. While our numerical results are encouraging, they are by no means definitive. More experiments are needed with other waveform data, including unlabeled data. Another avenue of pursuit is the theory: is SBD a metric? Is there any convergence theory for k-Shape, FCS+ or FCS++? Moreover, any satisfactory convergence theory has to overcome the fact that the SBD and SE schemes are computer programs, not functions.

REFERENCES

[1] C. Ding, X. He, H. Zha, and H. D. Simon, "Adaptive dimension reduction for clustering high dimensional data," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 147–154.

[2] E. Keogh, "A decade of progress in indexing and mining large time series databases," in *Proceedings of the 32nd International Conference on Very Large Data Bases*. VLDB Endowment, 2006, pp. 1268–1268.

[3] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[4] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," *Data mining in time series databases*, vol. 57, pp. 1–22, 2004.

[5] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research," *Knowledge and Information Systems*, vol. 8, no. 2, pp. 154–177, 2005.

[6] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[7] W. Meesrikamolkul, V. Niennattrakul, and C. A. Ratanamahatana, "Shape-based clustering for time series data," in *Advances in knowledge Discovery and Data Mining*. Springer, 2012, pp. 530–541.

[8] B. Hu, Y. Chen, and E. J. Keogh, "Time series classification under more realistic assumptions." in *SDM*. SIAM, 2013, pp. 578–586.

[9] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints." SIAM, 2004.

[10] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[11] V. Niennattrakul and C. A. Ratanamahatana, "Shape averaging under time warping," in *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. ECTI-CON 2009.*, vol. 2. IEEE, 2009, pp. 626–629.

[12] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. ACM, 2011, pp. 177–186.

[13] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1855–1870.

[14] J. C. Bezdek, "Fuzzy mathematics in pattern classification," 1973.

[15] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "The ucr time series classification/clustering homepage," *URL= http://www. cs. ucr. edu/eamonn/time series data*, 2006.

[16] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation. i," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 29–43, 1995.

[17] R. J. Hathaway and J. C. Bezdek, "Switching regression models and fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 3, pp. 195–204, 1993.

[18] L. Bobrowski and J. C. Bezdek, "c-means clustering with the l 1 and l norms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 545–554, 1991.

[19] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[20] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment," *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, 2012.

[21] S. Theodoridis and K. Koutroumbas, "Pattern recognition–fourth edition, 2009."

[22] R. N. Dave and K. Bhaswan, "Adaptive fuzzy c-shells clustering and detection of ellipses," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 643–662, 1992.

[23] D. T. Anderson, J. C. Bezdek, M. Popescu, and J. M. Keller, "Comparing fuzzy, probabilistic, and possibilistic partitions," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 5, pp. 906–918, 2010.

[24] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[25] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[26] M. Meilă, "Comparing clusteringsan information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[27] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[28] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical association*, vol. 32, no. 200, pp. 675–701, 1937.