Random projections fuzzy c-means (RPFCM) for big data clustering

Mihail Popescu*, James Keller#
University of Missouri
*HMI Dept., #ECE Dept.
Columbia, MO, USA
{popescum, kellerj}@missouri.edu

James Bezdek, Alina Zare#
University of Missouri
#ECE Dept.
Columbia, MO, USA
{bezdekj, zarea}@missouri.edu

Abstract— **Many contemporary biomedical applications such as physiological monitoring, imaging, and sequencing produce large amounts of data that require new data processing and visualization algorithms. Algorithms such as principal component analysis (PCA), singular value decomposition and random projections (RP) have been proposed for dimensionality reduction. In this paper we propose a new random projection version of the fuzzy c-means (FCM) clustering algorithm denoted as RPFCM that has a different ensemble aggregation strategy than the one previously proposed, denoted as ensemble FCM (EFCM). RPFCM is more suitable than EFCM for big data sets (large number of points, n). We evaluate our method and compare it to EFCM on synthetic and real datasets.**

Keywords—big data; dimensionality reduction; fuzzy c-means clustering; FCM; ensemble FCM; random projections

I. INTRODUCTION

Biomedical applications such as physiological monitoring, imaging, and sequencing produce large amounts of high dimensional data that in turn require new data processing and visualization algorithms [1]. One possible solution to these problems is based on dimensionality reduction. Principal component analysis (PCA), singular value decomposition (SVD) and many feature extraction algorithms have been proposed for reducing the dimensionality of a set of n data points to allow for faster processing time [1]. Among these dimensionality reduction algorithms, random projection [RP] methods inspired by the Johnson - Lindenstrauss lemma [2] have a special place due do their intriguing simplicity and efficiency.

Clustering is a popular data processing approach that can be used both in data summarization and visualization tasks. In this paper we propose a new random projection version of the well-known fuzzy c-means (FCM) clustering algorithm [3], denoted as RPFCM. Ensemble versions of FCM, denoted as EFCM, based on repeating random projections of the original data have been previously proposed [4],[5]. RPFCM, has a different ensemble membership aggregation strategy from EFCM. RPFCM is more suitable than EFCM for big data sets (large number, n, of points in R^p) since the final memberships are obtained by clustering an n×cN matrix, where c is the number of clusters and N is the number of repetitions (cN << n), instead of a n × n matrix (as in EFCM) or a n × p collection for the original data matrix (cN<<p). Practically, each R^p point from the original data matrix is replaced by its memberships in the cN clusters of the random projections. The rest of the paper is organized as follows: in section II we describe the main idea of random projections, in section III we present EFCM and RPFCM, in section IV we show results obtained on several datasets and in section V we draw the conclusions of this work.

MP was supported in part by the NSF SHWB grant # IIS-1115956

II. RANDOM PROJECTION FUNDAMENTALS

All random projection algorithms are based on the Johnson – Lindenstrauss (JL) lemma [2] that states that, with a well-defined probability, a set of n points $X = \{\mathbf{x}_1, \ldots \mathbf{x}_n\} \subset \mathbf{R}^p$ (denoted as the "upspace") can be mapped to a set of points $Y = \{\mathbf{y}_1,\ldots,\mathbf{y}_n\} \subset \mathbf{R}^q$ (denoted as the "downspace"), where q<<p, such that the pairwise distances from X are almost preserved in Y. The original lemma has several extensions where the mapping, the degree of approximation and its probability are further specified [6][7][8]. From this prior work, it is not clear which type of random projection is more appropriate for clustering, though we have begun that exploration in [13]. In this paper we use the variant of the JL Lemma proposed by Achlioptas in [6] as follows.

Theorem 1. Let P be a set of n points in $\mathbf{R}^p$, represented as an n × p matrix X. Given ε, β > 0 let

$$q_0 = \frac{4+2\beta}{\varepsilon^2/2-\varepsilon^3/3}\log n\,.$$

For integers $q \geq q_0$, let R be a p × q random matrix with $R(i,j)= r_{ij}$, where $\{r_{ij}\}$ are independent random variables from either one of the following probability distributions:

$$r_{ij} = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases} \quad (1)$$

$$r_{ij} = \begin{cases} +\sqrt{3} & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \\ -\sqrt{3} & \text{with probability } 1/6 \end{cases} \quad (2)$$

If $Y = XR/\sqrt{q}$ is the projection matrix of the n points in $\mathbf{R}^q$, then with probability $p > 1-n^{-\beta}$ for all $i,j \in \{1, n\}$ we have:

$$(1-\varepsilon)\,||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||\mathbf{y}_i - \mathbf{y}_j||^2 \leq (1+\varepsilon)\,||\mathbf{x}_i - \mathbf{x}_j||^2\,.$$

Theoretically the JL bound, q_0, does not depend on the number of dimensions p of the original space. For example, for n = 1,000,000 the JL bound given by Theorem 1 for ε = β =

0.25 is q_0=2387 while for n=1000 is q_0=1194 which would seem to make the usefulness of the theorem questionable, since 1194 features is still a big number. However, one should not consider the JL bound as being cast in stone. For example, other variants of the JL lemma [7], compute the JL bound as $q_0=2\log(n)/\varepsilon^2$, which produces a q_0=221 for n=1000. However, in practice, it has been observed [13] that the JL bound often holds even in spaces with $q << q_0$. In a related paper [13], the authors introduced the term *rogue random projections (RRPs)*, defined as projections for which $q << q_0$. As we will see in section IV, using ensemble aggregation, rogue projections may produce reasonable results even for q=2! The previous (EFCM) and the proposed version of the random projection FCM (RPFCM) are described in the next section.

III. Aggregation Strategies for Ensemble of Clusterers

The main idea of the ensemble approach is to cluster N random projections $\{Y_i\}$ of the data matrix X, using FCM, resulting in N fuzzy partitions $U_i = FCM(Y_i)$, and then to aggregate the resulting membership matrices U_i, $i \in \{1,N\}$. There are two main aggregation strategies for the N U_i's: 1) aggregate similarity matrices $M_i=U_i^tU_i$ and 2) concatenate the N U_i's to produce "membership vectors". The (j, k)-th element of an (n×n) M_i matrix represents the similarity between points $\mathbf{x}_1$ and $\mathbf{x}_j$ based on their FCM memberships. Concatenating N (c×n) matrices by stacking them on the element dimension, results in a n×cN matrix, which is significantly smaller than M_i. EFCM uses strategy 1) while RPFCM uses strategy 2).

A. Ensemble aggregation using U^tU (EFCM)

An ensemble clustering approach to finding significant genes in microarrays was proposed in [4]. FCM was used to cluster microarray data N times, N=30. For each trial the data $X\subset\mathbf{R}^p$ (p~10,000) was projected into a space $Y\subset\mathbf{R}^q$ (q~300) using Theorem 1 and a membership matrix U_i(c×n) was computed using FCM with c clusters, where c is constant for all i. The projection matrix R was computed using a i.i.d N(0,1) Gaussian distribution. For each trial, a "level of agreement" matrix M_i was computed as $M_i = U_i^tU_i$. The aggregated agreement matrix, M, for all N tries was then computed by averaging as $M = \Sigma_{i=1,N}M_i/N$. A dissimilarity matrix D computed by subtracting each element of M from 1, and denoted by D=1-M, was formed. The final partition matrix was computed by clustering the row vectors in the distance matrix D=1-M using FCM with c clusters.

A similar approach is described in [5] where *expectation-maximization* (EM) clustering was used to produce a similarity matrix M_i between joint probability distributions, $P(k|i,\theta)$ of point i being in cluster k= 1,c under model θ. The aggregated distance matrix, D (obtained as above by averaging over N=30 tries) was clustered using the hierarchical clustering approach called *complete linkage* (CL) [14].

In EFCM each point in R^p is replaced by the average over N projections of the pair-wise similarities to all of the other n-1 points. Consequently, ideal applications for EFCM are those in which the number of samples n is small (say n < 1000) and the original dimension p of the upspace is large (say p > 10,000) which is the case of the microarray data where p = 10,000 - 20,000 genes are sampled from n = 100 - 200 individuals. However, as n grows, EFCM becomes intractable for big data, for example, with the nimber of samples n in the millions.

B. Random projection FCM with Aggregation by concatenation (RPFCM)

The main disadvantage of the EFCM approach is that it results in an n × n agreement matrix M, which can't be further processed or visualized if n is large. The concatenation-based method we propose (RPFCM) does the final FCM clustering on the row vectors of an n × cN matrix, which will be useful when cN << n and cN<<p. A possible application is text processing, where clustering n ~ millions of documents is represented by a bag of 2000-10,000 words (p = 2000-10,000).

We propose two versions of RPFCM: one in which the number of c clusters is assumed known (RPFCM***c***); and another one (RPFCM***v***) in which the number of clusters is randomly selected from $\{c_{min}, c_{max}\}$. RPFCM***c*** has the following steps:

RPFCM***c*** *Step 0.*

Choose the number of trials N, the dimension q of the random projection space, and c, the number of clusters to seek. Input is a set of n p-dimensional data points, X(n,p);

RPFCM***c*** *Step 1*.

At each $i \in \{1,N\}$, construct a random projection matrix R_i(p,q) using Eq. (1) or (2);

RPFCM***c*** Step 2.

At each $i \in \{1,N\}$, compute the projection $Y_i = XR_i / \sqrt{q}$;

RPFCM***c*** *Step 3*.

At each $i \in \{1,N\}$, compute the (c×n) membership matrix U_i using FCM on Y_i, $[U_i] = FCM(Y_i, c)$;

RPFCM***c*** *Step 4*.

Compute the (n×cN) concatenation of the N fuzzy partition matrices, $U_{con}= [U_1^t |\ldots| U_N^t]$, and then compute the final memberships and cluster centers by clustering the row vectors of the aggregated matrix, $[U_f] = FCM(U_{con}, c)$.

If the number of clusters is unknown, the standard approach in RPFCM***c*** *Step 4* would be to perform the final clustering over a range of values of c and then use a cluster validity index such as the Xie-Beni index [9] to select the most plausible number of clusters. However, to facilitate comparisons between RPFCM and EFCM in Section IV, we will use a slightly different *Step 4*, denoted as:

RPFCM***c**** *Step 4**:

Compute a dissimilarity matrix $D=1-U_{con}U_{con}^t$ and then use CL to compute the final crisp partition matrix, $U_f^*=CL(D)$.

RPFCM***v*** and RPFCM***v**** are the same as RPFCM***c*** and RPFCM***c**** with one exception. In RPFCM***c*** *Step 3*, $[U_i]= FCM(Y_i, c^{\#})$, where $c^{\#}$ is a random choice from $\{c_{min}, c_{max}\}$. The lower (c_{min}) and the upper (c_{max}) limits for $c^{\#}$ are picked such that they under- and over-estimate the possible number of clusters. One could also vary q for each RP trial, but we are not investigating this possibility here.

Two points of note: (i) RPFCM is more memory efficient than EFCM; and (ii) computing the similarity matrix M_i for each RP iteration makes EFCM significantly slower than RPFCM.

IV. Results

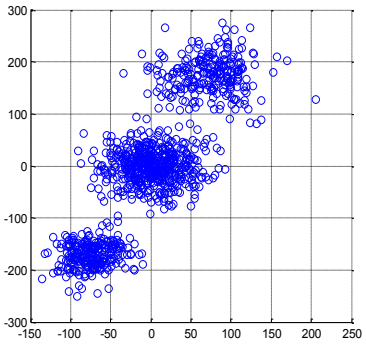
A. *Comparison of EFCM, RPFCM**c*** and RPFCM**v*** on two synthetic datasets*

To compare the EFCM, RPFCM***c**** and RPFCM***v**** algorithms we constructed two datasets: GM1 (a well separated Gaussian mixture) and GM2 (a tight Gaussian mixture). Both datasets have n=1000 data points from $\mathbf{R}^p$ with p =1000, generated from a Gaussian mixture with 3 components with mixing proportions $(p_1, p_2, p_3) = (0.25, 0.5, 0.25)$. The parameters for the two mixtures are shown in Table 1. Except for the very different means of component 1 and 3 of the two mixtures GM1 and GM2, $(2,2,\ldots 2)_{1000}$ vs. $(-2,-2,\ldots,-2)_{1000}$ and $(12,12,\ldots,12)_{1000}$ vs. $(-12,-12,\ldots,-12)_{1000}$, respectively, the other parameters were chosen at random.

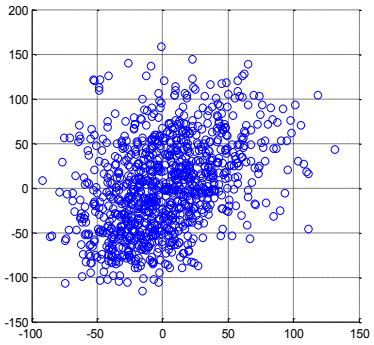
TABLE I. PROPERTIES (MEAN AND STANDARD DEVIATION) OF THE TWO DATASETS GM1 AND GM2

Component	1	2	3
	Mean		
GM2	$(2,2\ldots,2)_{1000}$	$(0, \ldots, 0)_{1000}$	$(-2,-2\ldots,-2)_{1000}$
GM1	$(12,\ldots,12)_{1000}$	$(0, \ldots, 0)_{1000}$	$(-12,\ldots,-12)_{1000}$
	Standard deviation		
GM2	$(1,\ldots,1)_{1000}$	$(2,2\ldots,2)_{1000}$	$(3,3\ldots,3)_{1000}$
GM1	$(1,\ldots,1)_{1000}$	$(2,2\ldots,2)_{1000}$	$(3,3\ldots,3)_{1000}$

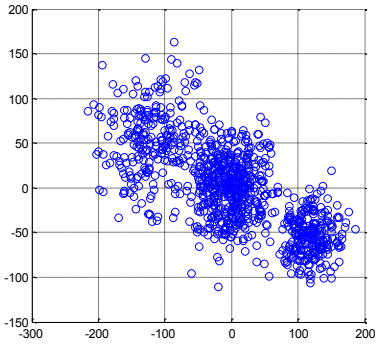
The JL bound for these data sets with the parameter choices $\varepsilon = \beta = 0.25$ is $q_0 \sim 1200$ (actually larger than the upspace dimension!). RRps are generated using Eq. (2). The scatter plot of a typical RRP in R^2 of each of the two datasets is shown in Fig. 1. Note that while GM1 looks well separated in Fig. 1(a), GM2.1 doesn't in Fig 1(b). However, we can find a projection (see Fig. 1(c)) where GM2.2 also looks somewhat separated.



(a) GM1



(b) GM2.1



(c) GM2.2

Fig. 1. Scatterplots of typical random projections of GM1 and GM2 in R^2.

The results of the EFCM, RPFCM***c**** (with *Step 4**) and RPFCM***v**** (with *Step 4**) for GM1 and GM2 are shown in Table II. We used N = 30 of random projections and a downspace dimension q = 2. To be able to compare the three algorithms we pre-generated the random projections and used the same projection matrices for all algorithms. The entire procedure was repeated 10 times. The comparison was performed by averaging the crisp Rand index [10] for the 10 trials between the generated partition U_f* using each of the three methods Recall that the final clusters are crisp because they are computed with complete linkage applied to dissimilarity matrices, and the ground truth partition is obtained from the generated Gaussian distributions in Table I.

TABLE II. AVERAGE RAND INDEX (300 TRIALS) AND STANDARD DEVIATION (*) FOR EFCM, RPFCM***c**** AND RPFCM***v**** ON GM1 AND GM2

	EFCM	**RPFCM*c****	**RPFCM*v****
GM1	0.9637 (0)	0.9637 (0)	0.963 (-0.002)
GM2	0.713 (0)	0.713 (0)	0.58 (-0.167)

Evidently RPFCM and EFCM are similar when the number of clusters is known (RPFCM***c****). When the number of clusters is unknown - the most likely the case in real applications - the performance of RPFCM***v**** is comparable to EFCM but has greater variability (the standard deviation for EFCM and RPFCM***c**** is virtually 0 while it is about 30% for RPFCM***v**** on dataset GM2. EFCM and RPFCM***c**** both experience degraded performance when applied to the more challenging dataset (GM2).

In Table III we show the average crisp Rand index obtained on GM2 for different numbers of RP repetitions, N. As expected, the mean Rand index improves as N increases, but RPFCM***c**** and EFCM level off beyond N=50. Consequently, the remaining experiments use N=50 trials. While increasing N is beneficial, it should be increased with caution since the dimension of the new clustering space is Nc and it should be kept <<p for a computation advantage to exist.

TABLE III. AVERAGE RAND INDEX FOR THE GM1 DATASET AND DIFFERENT NUMBERS (10N) OF N TRIALS

N	EFCM	RPFCMc*	RPFCMv*
10	0.3881	0.4532	0.4202
20	0.4860	0.465	0.4413
30	0.6480	0.7380	0.5826
40	0.7130	0.8383	0.7115
50	**0.9637**	**0.9637**	**0.7366**
100	0.9637	0.9637	0.8611

B. Experiments on the activity dataset, ACT [11]

To test our method with a real dataset, we chose the activity dataset [11, (https://archive.ics.uci.edu/)]. The dataset contains 19 activities such as sitting, standing, rowing and jumping captured by 45 sensors. The sensors, 3D accelerometer, 3D gyroscope and 3D magnetometer, were placed in 5 locations on the body (chest, legs and arms). Each activity is performed by 8 different subjects for 5 minutes. The sampling rate is 25 Hz, which produces a 337,500 dimension feature vector for each activity instance. To simplify our experiments, we only used 1 minute of activity, which resulted in a 152×67,500 data matrix (152 activity instances with 67,500 samples each). Our choice of activity length (1 min. segments) and raw data (as opposed to extracting features) was made, for convenience, to make computation more manageable.

The image of the ideal distance matrix, $I(D_{ACT})$, between all 152 activity instances is shown in Fig. 2(a) (diagonal blocks of size 8×8 with mutual distances 0, and distance values of 1 in the rest of the matrix). Similarly, we denote the ideal ACT partition as U_{ACT}, whose image looks similar to Fig. 2(a) but it has dimensions 19×152; dark =1 and white = 0 instead.

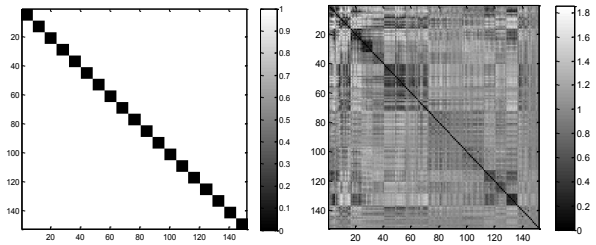
Since the values of the 45 sensors had different ranges and the same activities might be performed with some delay by different subjects we investigated different normalization and activity distance measures. We investigated three normalization procedures (no normalization, using the maximum for each feature, and statistical standardization by the mean and standard deviation of each feature) and four distance measures (Euclidean, scaled Euclidean, cosine and correlation). The most appropriate normalization and distance would result in a distance matrix image most similar to $I(D_{ACT})$.

To assess similarity with D_{ACT}, we computed the Pearson correlation $corr_P$ between D_{ACT} and the computed distance D, $corr_P(D_{ACT}, D)$. The $corr_P$ results for the three normalization procedures (no normalization-NO, division by maximum of each feature-MAX, subtraction of the mean followed by division by the standard deviation of each feature X, i.e. (X-mean(X))/std(X)-MEANSTD), and four distances (Euclidean, scaled Euclidean-sEuclidean, cosine and correlation) are shown in Table IV. For two feature vectors $x_1, x_2 \in R^{67500}$, the squared Euclidean distance was computed as $d_E2=(x_1-x_2)(x_1-x_2)^t$, the scaled Euclidean as $d_{sE}^2=(x_1-x_2)V^{-1}(x_1-x_2)^t$ with $V(j,j)=std(X_j)^2$, cosine as $d_c= 1-(x_1)^t x_2/(sqrt((x_1)^t x_1)sqrt((x_2)^t x_2))$ and correlation as the Pearson correlation of x_1 and x_2. The entire implementation was performed in Matlab using the *pdist* distance function with "euclidean", "seuclidean", "cosine" and "correlation" choices.

TABLE IV. $CORR_P$ BETWEEN D AND D_{ACT} FOR DIFFERENT NORMALIZATIONS AND DISTANCES

Feature normalization	Euclidean distance	sEuclidean distance	Correlation distance	cosine distance
NO	0.2228	0.2466	0.2332	0.2336
MAX	0.2794	0.2466	0.2399	0.2375
MEANSTD	0.2466	0.2466	0.4343	**0.4349**

None of the distances did very well (low correlations) with the first two normalizations, possibly due to temporal translations of the activities. The MEANSTD normalization (row 3 in Table IV) substantially improved the correlation between D and D_{ACT}. The image of the distance matrix $D_{MEANSTD,c}$ for the best case (bold in Table III) is shown in Fig. 2(b).



(a) $I(D_{ACT})$ (b) $I(D_{MEANSTD,c})$

Fig. 2. Images of the ideal ACT dataset (a); and one computed with the MEANSTD normalization and cosine distance (b)

While in Fig. 2(b) we can identify some activities that are very similar to themselves (the 17th activity, for example), the two distance matrices still look very different. This is obviously not an optimal solution, but in the rest of the experiments we will use the MEANSTD normalization and the cosine distance.

C. Comparison of EFCM, RPFCMc and RPFCMv* on ACT*

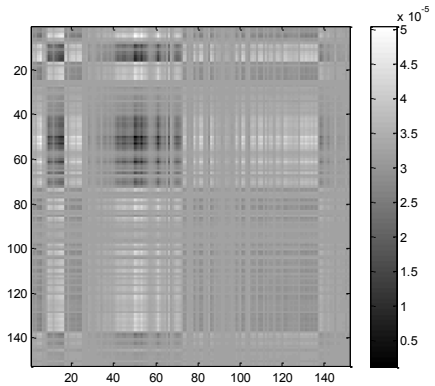
The JL limit for ACT with the same variables as in section IV.A is q_0=868. However, as we will see, we don't have to increase q too much above the lower limit of q = 2 to obtain reasonable clustering results. In Table V we show the average crisp Rand index obtained on the ACT dataset by the proposed clustering procedures for downspace dimensions q = {2, 10, 50, 200, 868}. Here, again, the RPFCM***v*/c**** procedures were run using *Step 4**. Hence the outputs are crisp 19-partitions of the 152 samples, and the crisp Rand index was computed as $Rand(U_f^*, U_{ACT})$. From Table V we see that RPFCM***v*/c**** and EFCM are comparable at most values of q.

TABLE V. AVERAGE RAND INDEX AND STANDARD DEVIATION (*) FOR 10 RUNS (500 TRIALS) OF EFCM, RPFCM***c**** AND RPFCM***v**** ON THE ACT DATASET

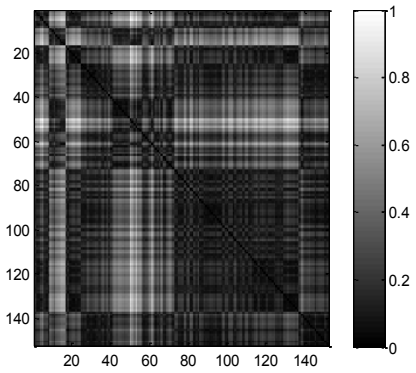
q	**2**	**10**	**50**	**200**	**868**
EFCM	0.26 (0.008)	0.258 (0.00004)	0.629 (0.0004)	0.629 (0.003)	0.63 (0.004)
RPFCM***c***	0.25 (0.035)	0.257 (0.011)	0.368 (0.101)	0.513 (0.085)	0.59 (0.091)
RPFCM***v***	0.26 (0.061)	0.269 (0.041)	0.463 (0.17)	0.587 (0.15)	0.634 (0.13)

EFCM seems to be less variable between the 10 runs (of N=50 RPs). As we see from table V, the EFCM standard deviation for 10 runs is at least an order of magnitude smaller than that of either RPFCM. This might be due to the EFCM aggregation process that happens at two levels: first, at the RP iteration level where the similarity between samples is computed as $U_i^t U_i$, and then at the ensemble level where the individual similarities are averaged. In RPFCM***v***/**c****, the cluster memberships assigned to each sample in each trial are concatenated without aggregation. Only in the final phase, *Step 4**, are they aggregated in the process of computing the dissimilarity matrix D.

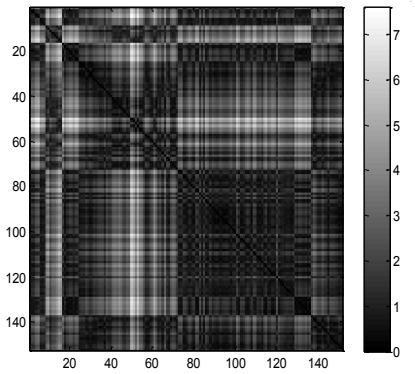
EFCM seems to reach its best performance at a lower RP number, around q=50, than either of the RPFCM approaches. From Table V we see that as q increases, RPFCM eventually catches up with EFCM. On the other hand, there are individual projections where RPFCM reaches higher Rand values than EFCM. For example at q=200, RPFCM***v**** reached a maximum Rand index of 0.76, RPFCM***c**** had a maximum of 0.75, while EFCM results were virtually identical for all 10 iterations (0.63). In Fig. 3 we show typical instance of images of the dissimilarity matrices that were used to compute the partitions in Table V for q=10. We see that the dissimilarity matrices obtained in RPFCM***c***/**v**** (Fig. 3(b) and 3(c), respectively) seem to contain more cluster like details than the corresponding image of the matrix from EFCM (Fig. 3(a)).



(a) $I(D_{EFCM})$



(b) $I(D_{RPFCMc*})$



(c) $I(D_{RPFCMv*})$

Fig. 3. Images of typical dissimilarity matrices produced by EFCM (a), RPFCV***c**** (b) and RPFCM***v**** (c) at q=10.

In conclusion, from the aggregation point of view, the RPFCM***v***/**c**** results are comparable to those produced by EFCM. On the other hand, RPFCM***v***/**c**** has the advantage of clustering the row vectors in a much smaller (n×Nc) aggregated membership matrix U_f than the n×n matrix used by EFCM. Next we look at clustering the fuzzy partition matrix U_f with FCM using *Step 4* instead of complete linkage (*Step 4**).

*D. Comparison of RPFCM**c** and RPFCM**v** on ACT*

The results obtained on ACT by our proposed clustering algorithms using FCM in *Step 4* are shown in Table VI. The comparison was performed by computing the average fuzzy Rand index [12] between the produced partition U_f and the crisp ground truth partition U_{ACT}. In row 1, RPFCM***c*** uses c=19 in each RP trial and c =19 in *Step 4*. In row 2, RPFCM***v*** uses a variable $c^{\#}$ for each RP trial and c =19 in *Step 4*. In row 3 we showed results obtained by using the Xie-Beni cluster validity in *Step 4* for RPFCM***v***. The value of $c^{\#}$ in row 3 is the average across 10 runs. All FCMs were run with m = 2.

TABLE VI. AVERAGE (500 TRIALS) FUZZY RAND INDEX FOR $RPFCM_c$ AND $RPFCM_v$ ON ACT

q	**2**	**10**	**50**	**868**
RPFCM***c***	0.8996	0.9019	0.9013	0.9027
RPFCM***v***	0.9009	0.9032	0.9034	0.9068
RPFCMv+ Xie-Beni	0.9009 c_{mean}=19.9	0.9194 c_{mean}=20	0.9126 c_{mean}=18.6	0.8979 c_{mean}=19.1

The most remarkable fact shown by Table VI is that we obtained virtually the same performance for q = 2 as for the JL bound q = 868 (the slight decrease is probably due to the random initializations for FCM). We see that using *Step 4* instead of *Step 4** produces an increase in the performance of about 30% as measured by the fuzzy RAND Index. This is probably due to the different nature of the two clustering algorithms used in the last step (*Step 4** - uses a crisp algorithm vs. *Step 4* – a fuzzy one). Previous experiments on the ACT dataset have produced resubstitution error rates between 92 and 99% [11], so a 90% Rand index is reasonably close to those results, particularly good for an unsupervised approach. A possible reason for the difference in accuracy could be that that experiments in [11] used 5s activity segments and extracted various statistical features from the signal. Using a variable cluster number at each RP iteration (as in RPFCM***v***) and a Xie-Beni cluster validity measure [9] in *Step 4* provided the same results as if the cluster number was known (even 2% better in some cases). Also, note the estimated average number of clusters was very close to the number of physical labels in ACT (c=19).

V. CONCLUSIONS

In this paper we presented RPFCM, an ensemble approach to clustering in many random projections based on fuzzy c-means. RPFCM is more suitable for large datasets than the the previous ensemble approach called EFCM. We showed that RPFCM gives similar results to EFCM when the last step is crisp clustering (step 4*), while it uses less memory and it runs faster. We also showed that RPFCM can give reasonable results even for rogue projections (i.e. $q << q_0$) [13]. This is very important when the dataset has not only many samples but many features.

As seen from the, sometimes large variance of RPFCM, some projections are better than others. Since in real clustering problems c is never known, RPFCM***v*** seems like a particularly attractive approach. The question remains whether it is better to try to choose a good projection (see methods in [13]) and cluster the resulting projected data once, or to use an ensemble methodology and then aggregate the clustering results of multiple projections (with some being possibly "bad" ones).

ACKNOWLEDGMENT

MP would like to thank National Science Foundation for the support.

REFERENCES

[1] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data", *Proc. ACM SIGKDD*, 2003, pp. 245-250

[2] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, 1984, pp. 189-206.

[3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1981.

[4] R. Avogadri, G. Valentini, "Fuzzy ensemble clustering based on random projections for DNA microarray data analysis", *Artificial Intelligence in Medicine*, 45, 2009 173—183.

[5] X.Z. Fern, C.E. Brodley, "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach", *Proc. ICML*, Washington DC, 2003

[6] D. Achlioptas. Database-friendly random projections. *Proc. ACM Symp. on the Principles of Database Systems*, pages 274–281, 2001.

[7] J. Matousek, "On variants of the Johnson-Lindenstrauss lemma", *Random Structures & Algorithms*, Volume 33, Issue 2 (September 2008)

[8] Santosh Vempala. *The Random Projection Method*. American Mathematical Society, Providence, RI, 2004.

[9] X. L. Xie and G. A. Beni, "Validity measure for fuzzy clustering," *IEEE Trans. PAMI*, vol. 3, no. 8, pp. 841-846, 1991.

[10] W. M. Rand (1971). "Objective criteria for the evaluation of clustering methods". *Journal of the American Statistical Association*, 66 (336): 846–850. doi:10.2307/2284239. JSTOR 2284239.

[11] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, 43(10):3605-3620, October 2010.

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

[13] X. Ye, J. C. Bezdek, J. M. Keller, M. Popescu, A. Zare, "Rogue Random Projections: Exploring downspaces below the JL Limit", *IEEE Trans. PAMI*, 2015, under review.

[14] R.A. Johnson and D. A. Wichern, *Applied Multivariate Statistical Analysi*s, 6th Edition, Prentice-Hall, Englewood Cliffs, NJ, 2007.