kEFCM: kNN-Based Dynamic Evolving Fuzzy Clustering Method

Shubair Abdulla
Depart. of Instructional & Learning Technologies
Education Colloege
Sultan Qaboos University
Muscat, Oman

Amer Al-Nassiri
IT College
Ajman University of Science and Technology
Fujairah - Campus
United Arab Emirates, Fujairah

***Abstract*—Despite the recent emergence of research, creating an evolving fuzzy clustering method that intelligently copes with huge amount of data streams in the present high-speed networks involves a lot of difficulties. Several efforts have been devoted to enhance traditional clustering techniques into on-line evolving fuzzy able to learn and develop continuously. In line with these efforts, we propose kEFCM, kNN-based evolving fuzzy clustering method. kEFCM overcomes the problems of computational cost, dynamic fuzzy evolving, and clustering complexity of traditional kNN. It employs the least-squares method in determining the cluster center and influential area, as well as the Euclidean distance in identifying the membership degree. It enhances the traditional kNN algorithm by involving only cluster centers in making classification decisions and evolving on-line the clusters when a new data arrives. For evaluation purpose, the experimental results on a collection of benchmark datasets are compared against other well-known clustering methods. The evaluation results approve a good competitive level of kEFCM.**

Keywords—Evolving; Fuzzy Logic; Clustering; k-NN

I. INTRODUCTION

Clustering Analysis is broadly applied successfully in many research areas such as market research, pattern recognition, data analysis, image processing, and document categorization [1] [2] [3] [4]. Clustering aims at describing data by defining set of clusters, which are naturally circles, based on similarities. The approach of finding approximate centroids is commonly used to form the clusters. A cluster centroid is used to determine the cluster location, and later, the system will tell to which cluster a group of input vector belongs by measuring the similarity in predefined features. Forming the clusters also involves determining the influential area of the clusters, which is equal to the radius.

In clustering, there are two crucial terms: fuzzy and evolving. The fuzzy term refers to the overlapping in clusters that is each element in a dataset belongs to one or more cluster in a degree. The cluster belongingness, called fuzzy membership (μ_{ij}), is used to discover the relation between the data element and disclosed clusters. The Euclidean distance is employed commonly to obtain the fuzzy membership values of elements in different clusters, i.e. distance between data point and cluster center. Technically, the evolving term means ability of the system to dynamically updating the clusters, adjusting the clusters centers and/or radius, to accommodate new unseen data when presented.

Beside their ability of analyzing data and making decisions based on acquired intelligence, the evolving clustering methods play an essential role in fuzzy rule-based systems (FRBS) and neuro-fuzzy systems (NFS) which are intelligent systems able to learn and develop continuously in order to enhance their performance. Over the last decade, the evolving clustering methods has boosted the emergence of these systems [5].

Designing an evolving fuzzy clustering algorithm involves a lot of difficulties. In the present high-speed networks, the huge amount of data streams, such as IP flows and network payloads, calls for on-line, fast, non-iterative evolving methods. Dealing efficiently with huge amount of multi-dimensional data items can be problematic because of clustering complexity and computational cost. The algorithm has to perform an incremental learning paradigm that is carried out to update the knowledgebase whenever new data emerges. Moreover, it has to efficiently manage previously seen training data to accommodate new data, and that needs an efficient memory management mechanism.

Unfortunately, most of the data clustering techniques such as K-means [6], Fuzzy C-means, Mountain clustering, and Subtractive clustering [7] lacks these capabilities.

Recently, the issue of creating evolving fuzzy clustering approaches to obtain the best fit of a dataset has been the subject of several research efforts. The research trends may be broadly divided into two directions: (i) to invent new techniques; (ii) to enhance traditional clustering techniques.

The k-Nearest Neighbors (kNN) clustering method [8] is among the clustering techniques in which development has seen attempts. kNN is one of the most simple machine learning methods. It can be used as a baseline for large developmental expansions. It has been selected as one of the top 10 data mining algorithms [9]. However, despite these pros, it has some cons: (i) it is computationally expensive; (ii) it requires large memory; (iii) it does not have ability to learn which data are most important.

In line with the trends that seek to enhance traditional clustering techniques, we present an enhanced version of the kNN algorithm, kNN-based Evolving Fuzzy Clustering Method, kEFCM for short. It is worth mentioning that kEFCM is introduced as a preprocessor for the neural fuzzy inference model [10]. The problems of designing an evolving fuzzy clustering method are addressed through many enhancements

to the original kNN approach such as: reducing the complexity of computation, on-line clustering, and fuzzy evolving. To reduce the computational expense, kEFCM considers the cluster centers only in making classification decisions. The knowledgebase evolving is carried out simply by assigning the coordinates of a new coming example to a new cluster center, and the radius will be the arithmetic mean of all radiuses, in case the example does not belong to any cluster. Neither thresholds nor constraints have been used in the on-line phase.

The rest of this paper is organized as follows. In Section II, we discuss related work, and in Section III, we review the kNN algorithm. The kEFCM approach is explained in Section IV, and we report experiments on real dataset in Section V. Finally, Section VI concludes and indicates the directions for future work.

II. Related Work

Reviewing the literature yields a plenty of clustering approaches. Comprehensive surveys have been published on clustering such as Jiang et al. [11] Xu and Wunsch II [12], and Hruschka et al. [13]. Since they are the main subjects of this paper, we limited our revision to the approaches of evolving fuzzy clustering and to those approaches that are devoted to enhance the kNN clustering method.

The First attempts of data fuzzy clustering could date back to the last century. However, it is still an open problem especially in the present, vast amounts of online information exchange. k-Means clustering [14] is based on finding data clusters such that an objective function of distance (Euclidean distance in most cases) measure is minimized. This algorithm in non-fuzziness and does not solve the overlapping issue. It gives either 1 when a data belongs to a cluster or 0 otherwise. The fuzzy c-means (FCM) is the most popular fuzzy clustering algorithm that also uses an objective function while clustering the data. A given data may belong to several clusters in different digress identified by membership value from 0-1. Since it has a number of drawbacks such as high time requirements, noise, and difficulty in identifying the initial clusters [15], some developments have been suggested. One of these developments is the Possibilistic FCM (PFCM) [16] which is an attempt to solve the noise sensitivity defect of the FCM [13]. The Multi-Kernel Fuzzy Clustering (MKFC) [17] is another attempt to develop the FCM which addresses the problem of limitation to spherical clusters. It incorporates multiple kernels and automatically adjusts the kernel weights to make the system immune to ineffectiveness kernels and irrelevant features.

In 2002, Kasabov and Song [3] introduced the Evolving Fuzzy Clustering Method (ECM), which is considered as first evolving on-line clustering method [18]. ECM operates in two phases: off-line and on-line. In off-line phase, it estimates dynamically the number of clusters in a one-pass algorithm. The number of clusters depends on a threshold value, Dthr, which has to be tuned initially. The Dthr is used to control the maximum distance between a data and the cluster center. In the on-line phase, when ECM receives a data sample, based on its position in the dataset, ECM either creates a new cluster or updates some existing clusters. The value of Dthr is used to control updating cluster centers, if the radius equals to Dthr, the cluster will not be updated.

Some sophisticated fuzzy clustering methods have emerged over the past few years. For example, the Fuzzy Rule-Based Classifier (FRBC) [13] inherently performs the unsupervised cluster analysis by employing a supervised classification approach. It explores the potential clusters and identifies them by using interpretable fuzzy rules. The actual boundaries are revealed through simultaneous classification of data with the fuzzy rules. The Evolving Local Means (ELM) [19] is another example. It is simple and has the desirable features of density based approached. It uses the concept of non-parametric gradient estimate of a density function. The evolving process is performed based when the density pattern changes.

The research of deriving fuzzy clustering methods from the traditional kNN algorithm was initially motivated by its drawbacks. For example, the authors in [20] present a clustering ensemble algorithm based on kNN. To summarize the ensemble data, the algorithm generates the similarity matrix of data and then it uses hierarchical clustering to get the final clustering. Another example can be seen in [21]. It is a special cluster matching algorithm that establishes correspondence among fuzzy clusters by building a new combination model based on cluster matching and fuzzy majority vote. In [22], a new kNN-based clustering method, called kNNModel, is proposed. The model is similar to kNN, but the k value is automatically determined. A data model is built by extracting a set of representatives of the training data. The representatives whose size is far less than the whole training data are involved in making classification decision. The kNNModel is enhanced in [23] by developing a cluster-based training algorithm to learn the optimized set of representations.

In some sense, the performance of the reviewed evolving fuzzy clustering methods is effective. However, we believe that an effective clustering method should possess the features: (1) fuzzy clustering, (2) dynamic evolving, (3) low computational cost, and (4) little efforts for prior tuning. The demand of such method has not been yet achieved. For example, although the ECM is on-line evolving fuzzy clustering, its performance relies on prior precise tuning of Dthr parameter. With respect to the methods that aim at developing kNN algorithm, unfortunately, the dynamic evolving is still a crucial demand. The kEFCM approach is concern about the dynamically evolving which distinguishes it from the above mentioned development of kNN.

III. kNN: k-Nearest Neighbors Algorithm

In this section, we briefly describe the kNN algorithm. kNN is an instance-based learning algorithm. Although it is most often used for classification, it also can be used in estimation and prediction. Given a set of training data, a new data may be classified simply by comparing it to the most similar data in the training dataset. The process of building kNN classifier involves identifying k value, the number of the most similar classes to be considered in the training dataset. The process involves also measuring the similarity based on defining the distance function. The most commonly used distance function in Euclidean distance:

$$d_{Euclidean}(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \ldots\ldots \quad (1)$$

Suppose that we are interested in classifying the type of network packet captured by a traffic collector system based on certain characteristics, such as the payload size and the destination port#. For a sample of 200 packets, Figure 1 shows a scatter plot of the packet size against the destination port#. The type of the points symbolizes a particular network packet class. Circle points indicate A class; diamond points indicate B class; square points indicate C class.

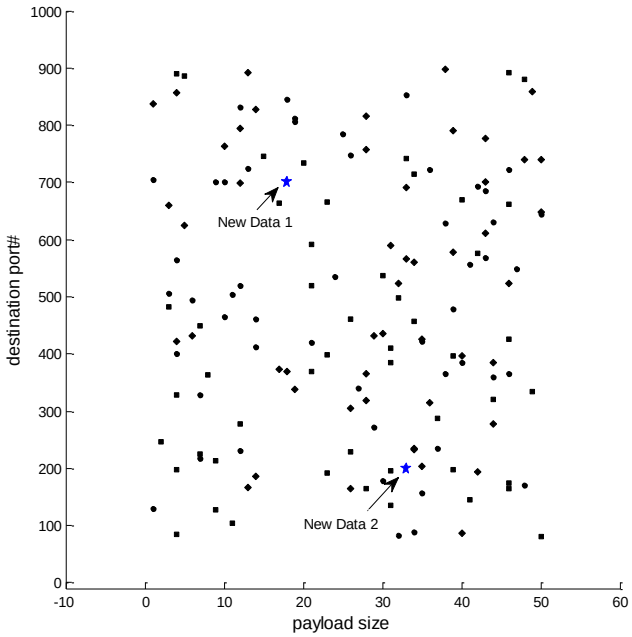

Fig. 1. Scatter plot of payload size against destination port#

Now suppose that there are two new network packets (indicated as stars in the Figure) without classification and would like to classify them based on other packets with similar attributes. New data 1 is composed of 18 bytes and directed to port# 700. Since the packet attributes place it into a section where six packets of the nearest packets belong to C class (square), we would thereby classify it as C easily.

Regarding the new packet 2, which is 33 bytes directed to port# 200, suppose k=1 so that any new data would be classified according to whichever one point it closest to. In this case the packet would be classified into B class since that the closest packet on the scatter plot belongs to B class (diamond point). Suppose we now set k=2 so that the new packet 2 would be classified according to the classification of 2 packets closest to it. One of these packets belongs to C class (square) and one belongs to B class (diamond). The kNN classifier cannot decide between these two classifications. The voting is helpless here since there is one vote for each of two classes. The voting will not help either for k=3 in case of the three nearest packets belong to three different classes.

After determining which training data are most similar to the new unseen data, we need to establish a combination function for classification decision. A combination function could be unweighted voting (each neighbor has one vote) or weighted vote (closer neighbors have larger vote). In either case, this function is computationally expensive.

The above example has shown that the number of nearest neighbors, k, is considered as one of the most influential factors in the accuracy of the classification. The value of k must be set carefully, small value may maximize the probability of misclassification, and large value may make the k nearest packets distant from the right class. The obvious best solution is to employ a cross-validation procedure which is done by trying various values of k with different randomly selected training datasets and determining precisely the k value that minimizes the classification error.

IV. kEFCM: kNN-Based Evolving Fuzzy Clustering Method

The kEFCM runs in two phases: off-line and on-line phase. During the off-line phase, kEFCM partitions the input space into clusters, while in the on-line phase; kEFCM classifies new coming data and updates dynamically the clusters for the purpose of evolving.

A. Off-line Clustering Phase

In the off-line phase, kEFCM applies fast, optimized technique for clustering dataset points. Figure 2 presents a high-level overview of kEFCM in the off-line clustering process.

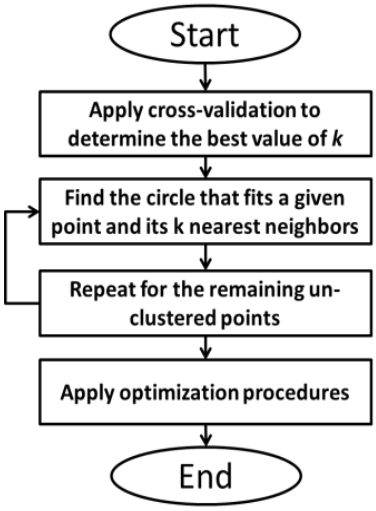

Fig. 2. kEFCM Off-line Clustering Phase

The process of off-line clustering starts by taking the first sample of the dataset (x_1, y_1) and finds its k-nearest points (X_i, Y_i) using the Euclidean distance (eq. 1) where $i = 1,2,\ldots,k+1$. Then the least squares method (LMS) is used to find the equation of the circle that best fits the points (X_i, Y_i) by calculating the center and the radius. A linearized model of the circle equation is needed to determine the values of center (a, b) and radius (r):

$$(x_i - a)^2 + (y_i - b)^2 = r^2 \ldots\ldots\ldots\ldots (2)$$

The linearized model of this equation:

$$x_i^2 - 2ax_i + a^2 + y_i^2 - 2by_i + b^2 = r^2$$

$$x_i^2 + y_i^2 = 2ax_i + 2by_i + r^2 - a^2 - b^2$$

$$x_i^2 + y_i^2 = Ax_i + By_i + C \ldots\ldots\ldots\ldots (3)$$

Equation (3) is now linear with three undetermined coefficients, A, B, and C. In this case, the matrices are used to solve the least squares problem:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & k+1 \end{bmatrix}^{-1} \begin{bmatrix} \sum x_i (x_i^2 + y_i^2) \\ \sum y_i (x_i^2 + y_i^2) \\ \sum x_i^2 + y_i^2 \end{bmatrix} \ldots\ldots\ldots\ldots (4)$$

After having values for A, B, and C, the circle is simply determined by calculating its center (C_x, C_y) and radius (r):

$$C_x = \frac{A}{2} \ldots\ldots\ldots\ldots (5)$$

$$C_y = \frac{B}{2} \ldots\ldots\ldots\ldots (6)$$

$$r = \frac{\left(\sqrt{4c + A^2 + B^2}\right)}{2} \ldots\ldots\ldots\ldots (7)$$

This process is repeated on the remaining data points. As this step may create unwanted overlapped clusters, the next step applies an optimization procedure that handles two constraints: (1) The number of clusters that contain small cluster(s) is equal to 0; (2) The number of clusters that include points less than k is equal to 0. These constraints are represented mathematically by two functions: probability of inclusion P(I) and probability of violation P(V) respectively:

- Probability of inclusion *P(I)*

$$P(I) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} f(C_i, C_j)}{n(n-1)/2} = 0 \ \forall\, i \neq j \ldots\ldots\ldots\ldots (8)$$

Where:

$f(C_i, C_j)$: the inclusion function:

$$f(C_i, C_j) = \begin{cases} 1 & C_j \subset C_i \\ 0 & C_j \not\subset C_i \end{cases}$$

C_i, C_j *:* any cluster, n: # of clusters

- Probability of violation P(V)

$$P(V) = \frac{\sum_{i=1}^{n} f(C_i)}{n} = 0 \ \ldots\ldots\ldots\ldots (9)$$

Where:

$f(C_i)$ *:* violation function,

$$f(C_i) = \begin{cases} 1 & C_i < k \\ 0 & otherwise \end{cases}$$

C_i*:* any cluster, n: # of clusters, k: # of *nearest neighbors* $k < n$

B. Algorithm of kEFCM Off-line Phase

The kEFCM off-line clustering algorithm is given below as pseudo code:

INITIALIZATION:
N: No of samples,
n <= N : No of prototype samples,
k: No of nearest neighbors.
BEGIN %Off-line phase
Step 1: Take a data sample and find its k-nearest samples by using Euclidean distance.
Step 2: By using equations (4), (5), (6), and (7), determine the cluster that fits the sample and its k-nearest samples.
Step 3: Repeat steps 1 and 2 for remaining samples.
Step 4: Find the probability of inclusion P(I) and probability of violation P(V) for the partitioning by using equations (8) and (9).
Step 5: If P(I)=0 and P(V)=0 then STOP
Step 6: Else, remove all inclusion:
$\forall f(C_i, C_j) = 1$
Set $C_i = C_i \cup C_j$
And adjust the # of violated neighbors to $\geq K$
Step 7: Go to Step4
END %Off-line phase

Figures 3-5 explain graphically four cases that possibly happen during the off-line clustering phase, assuming that the total number of staring points is 19 and the optimum value of k is 3.

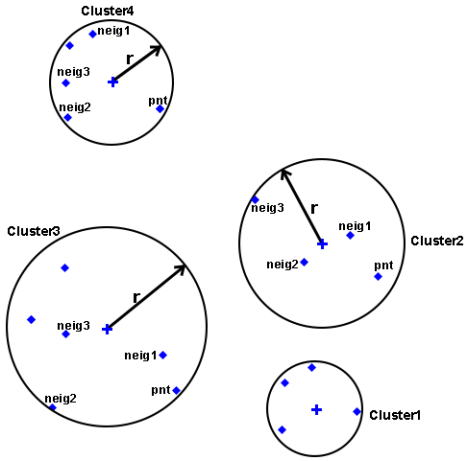

Fig. 3. Case (A): this case is normal case. In cluster 3, the point received by kEFCM is "pnt" and its three neighbors (nieg1, nieg2, and nieg3) have formed a valid cluster. Two points (unlabeled points) will be included in the cluster as they are placed within the cluster influence range

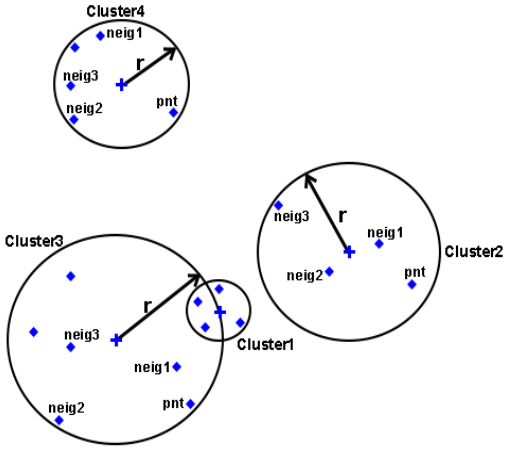

Fig. 4. Case (B): despite that cluster 1 overlaps cluster 3, no optimization is need as no cluster contains small clusters (P(I)=0)

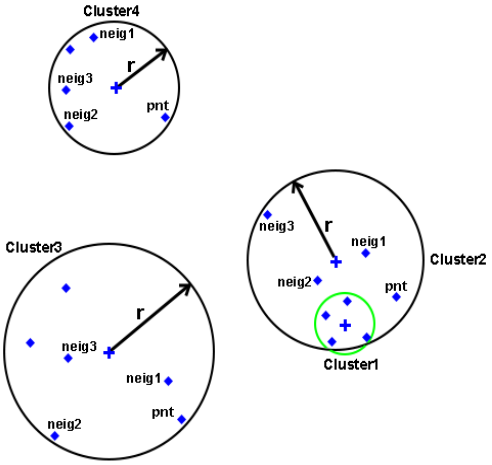

Fig. 5. Case (C): this case represents inclusion violation, cluster 2 contains cluster 1 (P(I)>0)

C. On-line Evolving Phase

This phase of kEFCM classifies new coming data and evolves the clusters dynamically. Figure 6 presents a high-level overview of the on-line process.

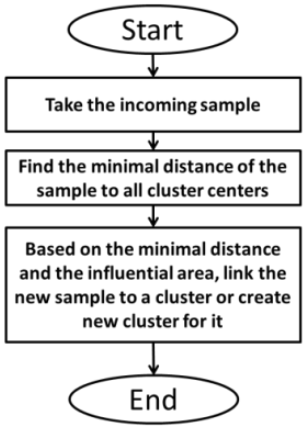

Fig. 6. kEFCM On-line Evolving Phase

The process of evolving operates on the partitioning model that resulted from the off-line phase. Whenever a new data example x is presented to the system, kEFCM updates the clusters according to the position of x. To reduce the complexity inherited from the kNN method, kEFCM computes the distance of x to the cluster centers only instead of computing the distance to all points. If x lies inside the influential range of a cluster, then kEFCM attaches it to the cluster and outputs the cluster class. Otherwise, a new cluster is created by simply assigning the coordinates of x to the center whilst the radius will be the arithmetic mean of all radiuses. The output class in this case is decided based on weighted voting combination function by considering the 3-nearest centers rather than data points.

Any new cluster created dynamically is updated if the number of its points reaches 3, and thereupon, center and radius of the circle that fits the 3 points are calculated by using LSM.

D. Algorithm of kEFCM On-line Phase

The following Algorithm summarizes the evolving process:

INITIALIZATION:
Take the partitioning resulted from off-line phase
BEGIN %On-line phase
Step 1: Take incoming sample x if available
Step 2: find the minimal distance (m) of x to the existing centers. Set the nearest cluster to Cm and its center to Cnm.
Step 3: If m<=Rm, then link the x sample with Cm. output the class of Cm.
a) If the # of samples in Cm =3 then update Cm
Step 4: Else,
a) Create a new cluster Ci+1
b) Center of the cluster Cni+1 = x
c) Calculate the mean of centers M, set Ri+1=M
d) Output the class of the 3-nearest centers by using the weighted voting
Step 5: Go to Step1
END %On-line phase

V. Experiments

This section describes the kEFCM evaluation process. Three sets of experiments were conducted to examine clustering quality, performance, as well as complexity and computational cost. For each set of experiments, we describe the measuring metrics, benchmarking algorithms, and the results of comparison. Before going through these parts, the datasets involved in the evaluation process are described and the results of cross-validation technique used to get optimum value of k are presented.

A. Dataset Used

To assess the quality of clustering of kEFCM, 6 datasets are used in the experiments, 1 forecasting dataset, the gas-furnace [24] and 5 classification datasets selected from KEEL Dataset Repository [25] and UCI Machine Learning Repository [26]. Table 1 summarizes the features and classes of these datasets.

As discussed in Section III, the choice of k is critical. To estimate the value of k accurately, the N-fold cross-validation technique is used. This technique involves setting aside some part of dataset elements for training and the rest for testing. The 10-fold cross-validation has been adopted throughout the experiments. First, the dataset is split into 10 folds. Then, the kEFCM is trained with 9/10 of the dataset, while the reminder 1/10, randomly selected one fold, is used for testing. Five values for k have been suggested: 3, 5, 7, 11, and 13.

TABLE I. Datasets Used for the kEFCM Evaluation

Dataset	Features	Classes	Samples
Gas-furnace	2	-	296
Iris	4	3	150
Glass	9	6	214
Ecoli	7	8	336
Balance Scale	4	3	625
Pima Indian Diabetes (PID)	8	2	768
Heberman Survival (HS)	3	2	306
Relation Banana (RB)	2	2	5292

Eventually, the k value that performed at the highest level of accuracy has been adopted. For the gas-furnace dataset, the k=13 is assigned manually as it has no classes. Table 2 summarizes the results obtained for each value over the dataset used.

TABLE II. N-FOLDS CROSS-VALIDATION RESULTS (%)

Dataset	k-values				
	3	5	7	11	13
Iris	82.9	83.1	**84.1**	84.0	81.8
Glass	88.1	88.1	87.9	88.2	**88.5**
Ecoli	92.7	93.1	94.2	**94.5**	91.8
Balance Scale	83.1	83.5	82.9	83.3	**83.6**
PID	96.4	97.2	97.0	**98.5**	95.8
HS	91.5	92.2	93.6	**94.2**	90.9
RB	61.1	60.5	62.3	62.3	**62.8**

B. *Clustering Quality*

Two parameters were taken as criteria in the comparative analysis:

- **MaxD**: the maximum distance between a point and its cluster center.
- **Cluster Purity:** The quality of cluster:

$$purity = \frac{\sum_{i=1}^{C} \frac{N_i^d}{N_i}}{C} \times 100\% \dots\dots\dots\dots (10)$$

Where C is total clusters, N_i^d is the number of members of the majority class in clusters i, and N_i is the total number of members in cluster i.

The clustering results were compared with those resulted by ECM on-line and off-line. Three experiments were conducted to examine the quality of clustering. In the first experiment, by setting the k value to 13, the kEFCM was employed to cluster the gas-furnace dataset into 15 clusters,

Figure 7 shows the clusters graphically. The results of the second experiment are show in Figure 8 which compares the MaxD values obtained by kEFCM and other clustering methods. In the third experiment, Figure 9, the kEFCM approach (k=7) was used to partition the Iris dataset. By using (eq. 10), the cluster purity is computed for kEFCM and other clustering methods.

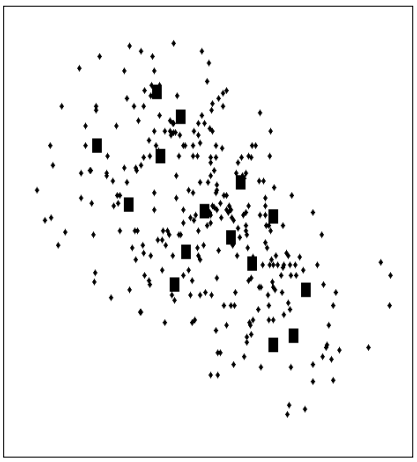

Fig. 7. Clustering gas-furnace dataset into 15 clusters (k=13), ♦: input vector, ■: cluster center

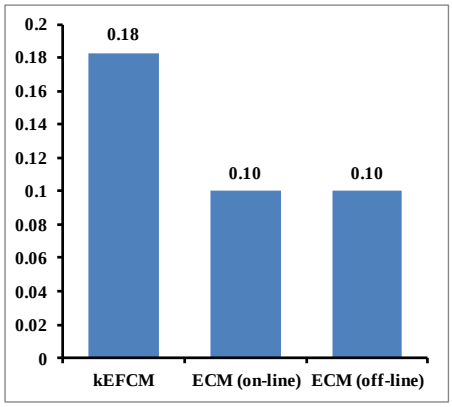

Fig. 8. Comparing the kEFCM against ECM in Terms of MaxD Over Gas-furnace Dataset

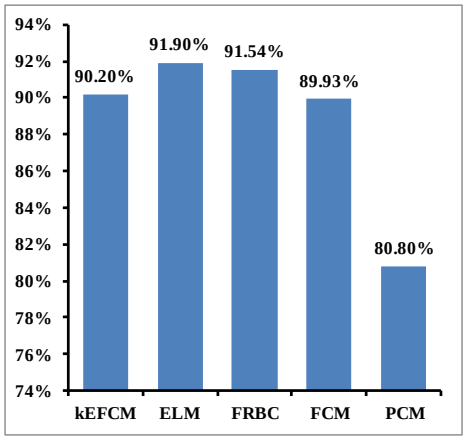

Fig. 9. Comparing the kEFCM against Fuzzy Clustering Methods in Terms of Clusters Purity Over Iris Dataset

TABLE III. COMPARING THE KEFCM AGAINST FUZZY CLUSTERING METHODS IN TERMS OF NMI OVER MULTIPLE DATASETS

Methods	Datasets				Average
	Glass (k=13)	Ecoli (k=11)	Balance Scale (k=13)	PID (k=11)	
kEFCM	0.341	**0.601**	**0.175**	0.116	**0.308**
k-Mean	0.320	0.570	0.121	0.102	0.278
FCM	0.333	0.574	0.118	0.114	0.285
MKFC	**0.355**	0.574	0.120	**0.140**	0.297

TABLE IV. COMPARING THE KEFCM AGAINST FUZZY CLUSTERING METHODS IN TERMS OF ARI OVER MULTIPLE DATASETS

Methods	Datasets				Average
	Glass (k=13)	Ecoli (k=11)	Balance Scale (k=13)	PID (k=11)	
kEFCM	0.177	0.384	**0.139**	0.140	0.210
k-Mean	0.172	0.384	0.129	0.136	0.205
FCM	**0.181**	**0.387**	0.138	**0.143**	**0.212**
MKFC	0.179	0.383	0.135	0.116	0.203

The following points can be concluded from the results:

- Although the perfect value of MaxD was obtained by ECMs, kEFCM achieved very close value, 0.180.
- We computed the standard deviation (stdev) of MaxD for all gas-furnace clusters to check the consistency in the size of clusters. The obtained value 0.3221 shows good consistency.
- Despite the fact that kEFCM is a single distance-based method and may create large number of unstable-mixed-class clusters [27], its ability to remove this kind of clusters is an advantage. kEFCM is equipped with optimization procedure that mainly works against unwanted clusters. It handles two constraints: P(I)=0 and P(V)= 0. The cluster purity reflects this ability, in contrast with ELM and FRBC, kEFCM performed at a comparative value 90.20%, which means that kEFCM produces small rates of unstable-mixed-class clusters. Also, kEFCM outperformed both FCM and PCM clustering methods.

C. kEFCM Performance

We used two common performance metrics to examine the kEFCM in terms of overlapping: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) [28]. These metrics compute the level of similarity between clustering system resulted by a method and compare it against the ground truth classes. A higher value means better clustering. There values ranged from 0-1, 1 means perfect match.

Equation 11 is used to compute the NMI of two clustering: C1 clustering resulted and C2 ground truth clustering, of a dataset X of n objects:

$$NMI(C1, C2) = \frac{I(C1, C2)}{\sqrt{H(C1)H(C2)}} \dots\dots\dots\dots (11)$$

Where:

$I(C1, C2)$ the mutual information between *C1* and *C2.*

H(C1) and *H(C2)* the entropy of *C1* and *C2*.

Regarding the ARI, the following equation is used:

$$ARI = \frac{a - \frac{(a+b)(a+c)}{a+b+c+d}}{\frac{(a+b)+(a+c)}{2} - \frac{(a+b)(a+c)}{a+b+c+d}} \dots\dots\dots\dots (12)$$

Where:

a = # pairs of data that are in the same class in $C1_i$ and same cluster in $C2_j$,

b = # pairs of data that are in the same cluster in $C2_j$, but not the same class in $C1_i$,

c = # pairs of data in the same class in $C1_i$, but not the same cluster in $C2_j$, and

d = # pairs of data that are not in the same cluster in $C2_j$ nor class in $C1_i$.

In Tables 3 and 4, we present the NMI and ARI values over multiple datasets for kEFCM and different methods. The last column (Average) of Table 3 displays the average of NMI value for each method over 4 datasets. kEFCM has the best average NMI over all methods. For each individual dataset, kEFCM outperforms all clustering methods in two datasets (Ecoli and Balance Scale), while in the other two datasets (Glass and PID), it is only outperformed by MKFC to be ranked as the second best method. Table 4 presents the results in terms of ARI. The results are slightly changed in contrast to NMI. The kEFCM is the second best in terms of average ARI. It is ranked first for only one dataset (Balance Scale) and ranked second for the remaining datasets. However, despite that, kEFCM, in overall results, has yielded a comparable stable performance.

D. Computational Time & Clustering Complexity

As discussed in Section IV, initially, kEFCM takes a data points and searches for a cluster that best fits the point with its k nearest data points. Then, it loops through the rest of the un-clustered points, each iteration of the loop repeats the same process. To prevent the unwanted overlapping clusters, kEFCM applies equations 8 and 9.

This set of experiments is devoted to examine the computational time along with complexity of the cluster resulting. For the purposes of comparing, we chose FRBC and FCM clustering methods. The results on Iris, Glass, and Ecoli datasets, which are appeared in related research papers, are

compared with those obtained by kEFCM in Table 5. Although the kEFCM consumes more computational time than FRBC and FCM, it is obvious that there is a few timing differences. Add to this, the computational time depends directly on the number of samples within the dataset.

TABLE V. COMPUTATIONAL TIME (SEC) OF KEFCM, FRBC, AND FCM

Methods	Datasets		
	Iris (k=13)	Glass (k=7)	Ecoli (k=11)
kEFCM	1.330	1.410	1.472
FRBC	1.000	1.000	1.200
FCM	0.170	0.100	0.100

With respect to the clustering complexity, according to our view, the complexity of clustering means:

1) Creation of a large number of clusters in off-line phase.

2) Generation of clusters is increasing exponentially in on-line phase.

It has been noted that the number of clusters highly depends on k value, the number of nearest neighbors, which means that the number of clusters is subject to control by the user. Any value of k gives a highly accurate result has to be adopted, since the accuracy is the most important criterion. However, in general, kEFCM shows adequate stability and constant evolution throughout the testing. Figure 10 illustrates two examples of cluster evolutions on different datasets. In the first example of the Heberman survival dataset (k=11), 6 clusters were created off-line to accommodate 10 samples. In on-line phase, when new 90 samples were introduced, it created 14 new clusters to accommodate them. In the second example, the Relational Banana dataset (k=13), kEFCM created only 5 clusters off-line to accommodate 10 samples, and then it created 24 clusters on-line to accommodate new 90 samples. Despite that kEFCM started in both examples with a big number of clusters, it created a very small number of clusters in on-line phase, which means, also, an effective way in clustering unseen samples dynamically.

VI. CONCLUSION

We have proposed in this paper kEFCM, kNN-based evolving fuzzy clustering method. It is an enhanced version of traditional kNN machine learning. kEFCM approach uses the least-squares method for determining the cluster center and radius. The Euclidean distance is used to reflect the membership of a data point in a cluster. The method performs an optimization procedure that handles two constraints, probability of inclusion P(I)=0 and probability of violation P(V)=0. In on-line phase, kEFCM is able to carry out the incremental learning, which is the core tool of evolving. It reduces the computational time that is inherited from kNN by involving the cluster centers in making classification decision.

The clustering ability of kEFCM was examined by benchmarking a collection of real-world datasets. The results obtained were compared against several well-known clustering methods. The results showed that the kEFCM performs at a good competitive level. The possible future work will turn to deploying kEFCM onto real-world environment, where intuitively, it will perform at the same level of success.

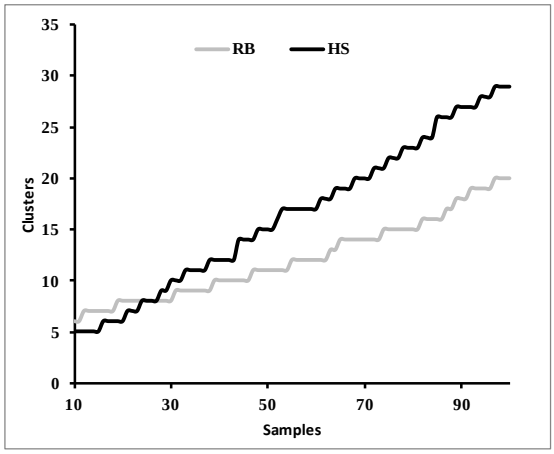

Fig. 10. Cluster Evolutions

REFERENCES

[1] F. Nie and P. Zhang, "Fuzzy Partition and Correlation for Image Segmentation with Differential Evolution," IAENG International Journal of Computer Science, vol. 40, pp. 164-172, 2013.

[2] R. C. D. A. K. Jain, Algorithms for Clustering Data: Prentice Hall, 1988.

[3] G. Mecca, S. Raunich, and A. Pappalardo, "A new algorithm for clustering search results," Data & Knowledge Engineering, vol. 62, pp. 504-522, 2007.

[4] A. K. Abd-Elaal, H. A. Hefny, and A. H. Abd-Elwahab, "Forecasting of Egypt Wheat Imports Using Multivariate Fuzzy Time Series Model Based on Fuzzy Clustering," IAENG International Journal of Computer Science, vol. 40, pp. 230-237, 2013.

[5] N. K. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," Fuzzy Systems, IEEE Transactions on, vol. 10, pp. 144-154, 2002.

[6] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," Systems, Man and Cybernetics, IEEE Transactions on, pp. 580-585, 1985.

[7] P. K. J. a. S. Chattopadhyay, "Comparative Study of Fuzzy k-Nearest Neighbor and Fuzzy C-means Algorithms," International Journal of Computer Applications, vol. 57, p. 10, November 2012.

[8] D. W. Aha, "Editorial," Artificial Intelligence Review, vol. 11, pp. 7-10, 1997.

[9] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, and S. Y. Philip, "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, pp. 1-37, 2008.

[10] A. Shubair, S. Ramadass, and A. A. Altyeb, "kENFIS: kNN-based evolving neuro-fuzzy inference system for computer worms detection," Journal of Intelligent and Fuzzy Systems.

[11] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," Knowledge and Data Engineering, IEEE Transactions on, vol. 16, pp. 1370-1386, 2004.

[12] R. Xu and D. Wunsch, "Survey of clustering algorithms," Neural Networks, IEEE Transactions on, vol. 16, pp. 645-678, 2005.

[13] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. P. L. F. De Carvalho, "A survey of evolutionary algorithms for clustering," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 39, pp. 133-155, 2009.

[14] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, pp. 100-108, 1979.

[15] E. G. Mansoori, "FRBC: A fuzzy rule-based clustering algorithm," Fuzzy Systems, IEEE Transactions on, vol. 19, pp. 960-971, 2011.

[16] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy c-means clustering algorithm," Fuzzy Systems, IEEE Transactions on, vol. 13, pp. 517-530, 2005.

[17] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Multiple kernel fuzzy clustering," Fuzzy Systems, IEEE Transactions on, vol. 20, pp. 120-134, 2012.

[18] V. Ravi, E. Srinivas, and N. Kasabov, "On-line evolving fuzzy clustering," in Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on, 2007, pp. 347-351.

[19] R. Dutta Baruah and P. Angelov, "Evolving local means method for clustering of streaming data," in Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on, 2012, pp. 1-8.

[20] F. Weng, Q. Jiang, L. Chen, and Z. Hong, "Clustering ensemble based on the fuzzy KNN algorithm," in Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, 2007, pp. 1001-1006.

[21] C. sheng Li, Y. nan Wang, and H. dong Yang, "Combining Fuzzy partitions Using Fuzzy Majority Vote and KNN," Journal of Computers, vol. 5, pp. 791-798, 2010.

[22] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, ed: Springer, 2003, pp. 986-996.

[23] L. Chen, G. Guo, and S. Wang, "Nearest neighbor classification by partially fuzzy clustering," in Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on, 2012, pp. 789-794.

[24] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," Physical review letters, vol. 59, p. 845, 1987.

[25] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," Journal of Multiple-Valued Logic and Soft Computing, 2010.

[26] A. Asuncion and D. J. Newman, "UCI machine learning repository," ed, 2007.

[27] E. Lughofer, Evolving fuzzy systems-Methodologies, advanced concepts and applications: Springer, 2011.

[28] J. V. de Oliveira and W. Pedrycz, Advances in fuzzy clustering and its applications: Wiley Online Library, 2007.