Incremental Clustering of Time-Series by Fuzzy Clustering

SAEED AGHABOZORGI, MAHMOUD REZA SAYBANI AND TEH YING WAH
Department of Information Science
University of Malaysia
Kuala Lumpur, 50603 Malaysia

Today, analyzing the user’s behavior has gained wide importance in the data mining community. Typically, the behavior of a user is defined as a time series of his or her activities. In this paper, users are clustered based upon time series extracted from their behavior during the interaction with given system. Although there are several different techniques used to cluster time series and sequences, this paper will attack the problem by utilizing a novel incremental fuzzy clustering strategy in order to achieve the objective. Upon dimensionality reduction, time series data are pre-clustered using the longest common subsequence as an indicator for similarity measurement. Afterwards, by utilizing an efficient method, clusters are updated incrementally and periodically through a set of fuzzy approaches. In addition, we will present the benefits of the proposed system by implementing a real application: Customer Segmentation. In addition to having a low complexity, this approach can provide a deeper and more unique perspective for clustering of time series.

Keywords: data mining, incremental clustering, time series, fuzzy clustering, prototype

1. INTRODUCTION

Currently, various data mining systems are being used to analyze data within different domains. In the context of data mining, clustering techniques are typically used in order to group data based upon their similarities. Most, if not all, developed clustering algorithms are compatible with static objects which their features values do not change or change insignificantly over time. On the other hand, unlike static objects, time series are classified as dynamic data which their features values are changing as a function of time. The behavior of users in interaction with various systems – like finances, healthcare, and business – is stored as historical data dynamically. As a result, this kind of dynamic data or, in another term, time series data is of interest for analysis.

There are several researches involving time series clustering, and there are different techniques used in time series clustering. However, in most of them, performance and accuracy of clusters are affected by new objects imported into the systems over time. Therefore, taking the future performance into account, complexity and accuracy of clustering in these systems should be our key focus. In these systems, extracted knowledge should be updated based upon new added objects or changes that take place in the existing objects over time. Without new arrived time series (e.g., if the number of bank customers does not change or new users do not drop by a website) or changes in on hand time series (if customers do not continue their activity in the website or bank), we could justify the usage of the traditional techniques for clustering time series systems. If, however, a system accepts new time series or changes to existing time series over time, then it should be updated in-
crementally.

Although, the power of processing has increased in recent years, nature of historical data stored as time series and the amount of data are big milestones for efficient clustering of time series. Another issue is limited space in memory to store entire time series data set in the memory for clustering. In addition, it would be an extra burden if we wanted to keep it up-to-date dynamically. Therefore there is a need for a dynamic clustering system with the ability of re-clustering time series data upon arriving new time series. Hence, the absence of an efficient methodology for clustering historical multidimensional data incrementally is felt. Here, a motivating scenario is described to demonstrate the issue requiring redressing. A large number of bank customers’ transactions are stored daily in databases on massive servers. In this case, the clustering of customers based upon their behavior forces banks to deal with a huge amount of multidimensional data. Based on observant projects in different banks, the data is usually converted to a lower dimension in a pre-processing phase for clustering purposes because of high computational costs. As mentioned before, it can be carried out by aggregation operations or transforming data. For example, in one of Malaysia’s banks, the classification of customers and defining the behavior of customers are used for different purposes like fraud detection or decision making in loan systems with different aggregated fields in use such as transaction count, maximum and minimum withdraw, deposit or fund transfer in a year. However, in most of these approaches, the quality of data was harmed and subsequently, the constructed model was not precise enough.

2. RELATED WORKS

Although there are variant researches and many projects relevant to analyzing time series data, clustering is one of the most frequently used techniques [1], due to its exploratory nature, and its application as a pre-processing phase in more complex data mining algorithms. Prior studies, projects and surveys that have noted different approaches and comparative aspects of time series clustering are listed as such: [2-12].

In this paper we focus on clustering of short-length time series and some of most popular and recent works in the literature are discussed here:

A two-factor fuzzy time series is presented in [13] to predict the temperature. Wang et al. combined high-order fuzzy logical relationships and genetic-simulated annealing techniques in their work. First, an automatic clustering algorithm is used to cluster the historical data into intervals of different lengths. Then, a method is developed to deal with the temperature prediction based on two-factor high-order fuzzy time series. They believe that the developed technique yields a higher average forecasting accuracy rate than the methods presented in previous works.

Cheng-Ping Lai et al., in [14], have adopted a two-level clustering method, where both the whole time series, and the subsequence of time series are taken into account. They, use SAX transformation in the first level to convert the time series into a symbolic representation and CAST to cluster level-1 data. Then, for calculating distances between level-2 data, DWT and Euclidean distance measurement is used for varying and equal length data respectively. Then similarity is defined as the inverse of this distance. At last, level-2 data, of all the time series, are then grouped by a clustering algorithm.
Another method to cluster time series is hierarchical clustering where nested hierarchy of similar groups is generated based on a pair-wise distance matrix of time series. Although, hierarchy clustering does not require the number of clusters as an initial parameter that is a well-known and outstanding feature of hierarchical clustering, the length of time series have to be identical due to the use of Euclidean distance as a similarity measurement [15]. Moreover, these algorithms are not able to deal effectively with long time series due to poor scalability [8].

Lin et al. present an anytime version of the partitioned clustering algorithm for time series in [16]. In this work, hierarchical and partitional clustering is used together with a symbolic representation of a numeric time series. In this method, they use the multi-resolution property of wavelets in their algorithm in such a way that the final centers obtained at the end of each level are reused as the initial centers for the next level of resolution. In [17] also, the same representation (symbolic representation of a numeric time series) is used together with a compression-based distance function.

Another kind of approaches that can be applied for time series analysis is based upon frequency-domain [18, 19]. The problem with these approaches is their weakness in reorganization of the structural features of a sequence; for instance, the place of local maxima or inflection points. Subsequently, it is difficult to find the similarity among sequences and produce value-based differences to use in clustering.

Hautamäki et al. in [20] focused on a problem pertaining to clustering of time series data in Euclidean space. They apply DTW (Dynamic Time Warping) and use the pair-wise DTW distances as input for next process. They use a Random Swap (RS) and Agglomerative Hierarchical clustering process in which \( k \)-means is used to fine-tune the output (compute locally optimal prototype). They believe that it provides the best clustering accuracy and also it provides more improvement to \( k \)-medoids. Considering the advantage of \( k \)-means which is a faster method than hierarchical clustering [21], the number of clusters has to be pre-assigned by user, which is not practical in obtaining natural clustering results. Additionally, \( K \)-mean is unable to deal effectively with long time series due to poor scalability.

### 3. INCREMENTAL CLUSTERING OF TIME SERIES DATA

Given the mentioned limitations in literature review, we present a novel method that is simple, flexible and accurate. In this research, the clustering is applied to time series in an incremental manner. At first, clusters are made in off-line mode. In order to achieve this objective, a subset of dimensionality reduced data is taken randomly, then the hierarchal clustering is applied on the data to estimate cluster numbers that best characterize the groups. Then incremental clustering is initialized after a period of time. This period can be characterized based on its application. For example, in a bank, it could be desirable to update a data warehouse (data set) with new transactions from customers on a monthly basis, whereas in other application such as web usage mining, we may consider changes in user behavior every year. In the incremental section, the algorithm focuses on assigning new time series to prepared clusters based on their similarity. That is, the cluster structure will be altered over time. Subsequently, the clusters are extended with the rest of the time series through a set of fuzzy approaches.
General speaking, incremental clustering is not a perfect approach for clustering because of its relatively low accuracy. However, time series data sets are quite large, and it is not possible to accommodate the entire data set in the main memory. In this situation, one approach is to store entire data matrix in a secondary memory and transfer time series to the main memory. In this method, only the cluster structure is stored in the main memory and the structure is updated incrementally to alleviate the space limitation. Using incremental approach is the main motivation in this paper.

The rest of this paper is organized as follows. In section 3, the terminology and methodology are described which include distance measurements and calculating prototypes. The algorithm is applied on real time series data sets and the experimental results are reported in section 4. In section 5, conclusions and achievements are drawn.

3.1 Dimensionality Reduction

One of the effective approaches utilized in many problems to reduce the dimension, is Discrete Wavelet Transform (DWT) [22-25], the discrete version of Wavelet Transformation. DWT is employed in wide range of applications from computer graphic to speech, image, and signal processing. The advantage of using DWT is its ability of representation of time series as multi-resolution. Additionally, the location of time and frequency can be gained by means of the time-frequency localization property of DWT.

Fig. 1. Coefficients of a one-dimensional signal constructed from ‘Haar’ wavelet decomposition structure.

3.2 Brief Review of Distance Measure Methods

In order to compare time series with irregular sampling intervals and length, it is of great significance to adequately determine the similarity of time series. There is different distance measurements designed for specifying similarity between time series. Zhang et al. [26] has performed a complete survey on different distance measurements comparing them in different applications. Dynamic time warping (DTW) [27, 28] is one of the most famous algorithms for measuring similarity between two sequences with irregular-lengths without any discretization which can be different in terms of time or speed. Similarly, LCSS (Longest Common Sub-Sequence) is a very useful measurement, especially for in front of noise and outliers in comparison with DTW because all points do not need to be matched in LCSS. Instead of a one-to-one mapping between points, a point with no good
match can be ignored to prevent unfair biasing. For aforementioned reasons in this research, LCSS is employed as distance measure for our methodology.

Longest Common Sub-Sequence: Given \( F_i \) as a time series and \( f^q_t \) as feature vector at time \( t \) in time series \( F_i \), if \( f^q_t \) is the feature \( q \)-th of time series for \( q = \{1, \ldots, p\} \) at time \( t \) and if \( p \) is number of features describing each object, then the LCSS distance is defined as [29]:

\[
\text{LCSS}(F_i, F_j) = \begin{cases} 
0, & T_i = 0 \quad T_j = 0 \\
1 + \text{LCSS}(F_i^{T_i}, F_j^{T_j}), & \text{otherwise}
\end{cases}
\]

where \( \text{LCSS}(F_i, F_j) \) value states the number of matching points between two time series and \( F_i = \{f_1, f_t\} \) specifies all the flow vectors in time series \( F_i \) up to time \( t \). In this paper, a customized distance measure is defined based on LCSS as:

\[
D_{\text{LCSS}}(F_i, F_j) = 1 - \frac{\text{LCSS}(F_i, F_j)}{\text{mean}(T_i, T_j)}
\]

where, using \( \text{mean}(T_i, T_j) \) instead of \( \text{min}(T_i, T_j) \) results in taking the length of both time series into account.

### 3.3 Brief Review of Fuzzy C-Means (FCM) Algorithm

One of the most extensively used clustering algorithms is the Fuzzy C-Means (FCM) algorithm presented by Bezdek [30]. FCM works by partitioning a collection of \( n \) vectors into \( c \) fuzzy groups and finds a cluster center in each group such that the cost function of dissimilarity measure is minimized. Bezdek introduced the idea of a “fuzzification parameter” \( m \) in the range \([1, n]\) which determines the degree of fuzziness (weighted coefficient) in the clusters [31]. Given \( c \) as number of classes, \( v_j \), centre of class \( j \) for \( j = \{1, \ldots, c\} \), \( n \) as the number of time series and \( \mu_{ij} \) as the degree of membership of the time series \( F_i \) to cluster \( j \) for \( i = \{1, \ldots, n\} \), distance of each time series \( F_i \) from each cluster center(\( v_j \)) is denoted as such:

\[
d_{ij} = (F_i, v_j) = D_{\text{LCSS}}(F_i, v_j).
\]

Let the centers be shown by \( v_j = \{v_1, \ldots, v_c\} \) and each time series by \( F_i \) that \( i = \{1, \ldots, n\} \) and \( d_{ij} \) as distances between centers and time series. Therefore, the membership values \( \mu_{ij} \) are obtained with:

\[
\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{1}{d_{ik}} \right)^{1/(m-1)}.}
\]

The FCM objective function (standard loss) that is attempted to be minimized takes
the form:

\[ J = \sum_{j=1}^{c} J_j = \sum_{j=1}^{c} \sum_{i=1}^{n} (\mu_{ij})^m d_{ij} \]  \hspace{1cm} (5)

where \( \mu_{ij} \) is a numerical value between \([0; 1]\); \( d_{ij} \) is the Euclidian distance between the \(j\)th prototype and the \(i\)th time series; and \( m \) is the exponential weight which influences the degree of fuzziness of the membership matrix.

In order to update a new cluster center value in iterations, the following formula is employed:

\[ v_j = \frac{\sum_{i=1}^{n} (\mu_{ij})^m (F_i)}{\sum_{i=1}^{n} (\mu_{ij})^m}, \forall j \in \{1, \ldots, c\}. \]  \hspace{1cm} (6)

### 3.4 Prototype Calculation Based on Existing Time Series inside Clusters

As it is mentioned previously, instead of defining random centers, random time series are clustered through hierarchal algorithm in pre-clustering phase. Afterwards, a time series is defined as prototype of each cluster which is reused as initial centre in next phase, incremental clustering. In this step, we use the Shortest Common Super Sequence (SCSS) to make the centroid.

Shortest Common Super Sequence: Given two sequences \( F_i = <f_{i1}, \ldots, f_{it}, \ldots, f_{im}> \) and \( F_j = <f_{j1}, \ldots, f_{jt}, \ldots, f_{jn}> \), a sequence \( U_k = <u_{k1}, \ldots, u_{km}, u_{km}> \) is a common super sequence of \( F_i \) and \( F_j \). The SCSS is a common super sequence of minimal length.

In the first step all dimensionally reduced time series are normalized. The following pictures illustrate the match points (Fig. 2) and the SCSS (Fig. 3) among three normalized time series in a sample cluster.

![Fig. 2. Match points between time series based on LCSS.](image1)

![Fig. 3. The calculated centroid of normalized time series.](image2)

In order to make the prototype based on the clusters’ members, a novel algorithm is presented in Table 1. In this algorithm, cluster \( C \) is denoted as \( C = \{F_1, \ldots, F_n\} \) and \( F_i \) as a member of cluster \( C \). For each \( F_i \) = \( \{f_{i1}, f_{i2}, f_{ip}\} \), \( f_0 \) is a point of time series \( F_i \) and \( \text{LCSS}(F_i, F_j)_{x \times 2} \) is the Longest common sub sequence of time series \( F_i \) and \( F_j \) and \( x \) is the total number of common points between them.
Table 1. Pseudo code for prototype calculation based on the LCSS.

<table>
<thead>
<tr>
<th>Initialize $n = \text{number of time series}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize $U$ with $P_{k1} = \text{LCSS}(F_1, F_2), P_{k2} = \text{LCSS}(F_1, F_2), 1 &lt; k &lt; \text{length}(\text{LCSS}(F_1, F_2))$</td>
</tr>
<tr>
<td>for all pair time series $i$ and $j$ $(i &lt; j)$:</td>
</tr>
<tr>
<td>$x \leftarrow 1; y \leftarrow 1; \quad$ while there are unvisited rows in $\text{LCSS}(F_i, F_j)$ do</td>
</tr>
<tr>
<td>initial a new pair $Q_x = &lt;Q_{x1}, Q_{x2}&gt;$ where $Q_{x1} = \text{LCSS}(F_i, F_j); Q_{x2} = \text{LCSS}(F_i, F_j)$;</td>
</tr>
<tr>
<td>Add $Q_x$ to $U$ in such a way that:</td>
</tr>
<tr>
<td>If there is not any pair include $P_{i, y}$ in $P_i$, then increase $y$;</td>
</tr>
<tr>
<td>else if $Q_{x1} &lt; P_{i, y}$ then insert $Q_x$ before $P_{i, y}$ in $U$; increase $x, y$;</td>
</tr>
<tr>
<td>else if $Q_{x1} == P_{i, y}$ then $P_{i, y} = Q_{x1}$; increase $x, y$;</td>
</tr>
<tr>
<td>else increase $y$;</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end while</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>for each $P_j$ in $U$:</td>
</tr>
<tr>
<td>$V_j = \text{mean}(f_{p_{j1}}, f_{p_{j2}}) \forall x, y$ which $&lt;p_{j1}, p_{j2}&gt; \in p_j$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>return $V_j$ as prototype of cluster $O$</td>
</tr>
</tbody>
</table>

3.6 Details of Methodology

There are 7 steps used in our methodology in order to cluster time series incrementally depicted in Fig. 4.

Step 1: Reduce dimension of time series
Step 2: Initialize Clusters
Step 3: Read new time series
Step 4: Identify time series which represents changes in cluster structure
Step 5: Create new cluster
Step 6: Assign the time series to clusters and move clusters
Step 7: Eliminate unchanged clusters

Fig. 4. Global view of methodology for incremental clustering.

Step 1: Reduce dimension of the time series. As mentioned in section 3.1, the reduced dimension time series are prepared using the gained wavelet decomposition. In this paper, Haar wavelet is recommended because it is the first and simplest wavelet technique. However, other wavelets such as Daubechies db1 also can be utilized for this purpose. For us-
ing Haar wavelet, the maximum level decomposition of a time series needs to be declared to avoid unreasonable level decomposition.

**Step 2:** Initialize clusters. Hierarchical clustering is used in this step to create initial clusters of the time series. Agglomerative techniques and LCSS are utilized as distance measure to applying hierarchical clustering. Although there are different methods for calculating distance (or similarity) between clusters, we use an unweighted average distance (UPGMA). In this approach, the distance between two clusters is measured by averaging distances between all pairs of time series where each pair is made up of one time series from each group. In this step we use the prototype that is made based on existing time series inside the clusters.

**Step 3:** Given \( m \) newly arrived time series in a cycle, \( k = \{n + 1, \ldots, n + m\} \) as index of time series, the distances between the new times series from existing clusters have to be computed to measure similarity. Because the clusters are represented by their prototype \( V_i \), distances between all prototypes and new time series are calculated as follows (Eq. (9)):

\[
d_{ij} = D_{LCSS}(F_i, V_j), \quad \forall j \in \{1, \ldots, c\} \text{ and } \forall i \in \{n + 1, \ldots, n + m\}. 
\]

Then we calculate the membership of new time series in all classes and then the membership matrix for time series gets updated:

\[
\mu_{kj}: \text{Membership of new time series } k, k = \{n + 1, \ldots, n + m\} \text{ to class } j, j = \{1, \ldots, c\}. 
\]

Then we find the most similar cluster to new time series based on the calculated membership.

**Step 4:** Identify whether new time series can be assigned to cluster directly or if it needs changes in cluster structure. To achieve this, we use the method that Crespo and et al. used in [32] to find outliers in a dynamic clustering. In order to determine if new time series are “not classified well by the existing classifier” and/or “it is far away from the current classes”, conditions one and two are checked on received data to detect such sort of time series (outliers):

**Condition 1** In this method, first of all, the distances between each pair of the prototypes have to be calculated. We use LCSS again to measure the distance between prototypes. The pair wise distance between existing clusters \( V_i \) and \( V_j \) is shown as:

\[
d(V_i, V_j), \quad \forall i \neq j, i, j \in \{1, \ldots, c\} 
\]

now we investigate:

\[
d_k > \frac{1}{2} \min \{d(V_i, V_j)\}, \quad \forall i \neq j, i, j \in \{1, \ldots, c\}. \]

This condition detects new time series that are far enough away from existing prototypes. That is, if the distance of a new time series is larger than half of the minimum dis-
INCREMENTAL CLUSTERING OF TIME-SERIES BY FUZZY CLUSTERING

distance of existing prototypes, we consider this time series as an outlier.

**Condition 2** The second condition is defined as such: Declare time series whose membership’s value is close to the inverse of the number of clusters.

\[
\mu_{kj} - \frac{1}{c} \leq \alpha, \quad j = 1, \ldots, c
\]  

(12)

Taking this condition into account, if the membership value of a time series is close enough to \(1/c\) (inverse of class number), then the time series is defined as an ineligible time series. The parameter \(\alpha > 0\) is considered as a closeness threshold for this condition.

At last, given two previous conditions applied on new time series, the outlier time series are defined as

\[
\text{Outlier}(F_k) = \begin{cases} 
1, & \text{if } F_k \text{ fulfills conditions 1 and 2} \\
0, & \text{else}
\end{cases}
\]

(13)

It shows the new time series which cannot be assigned to existing clusters if Outlier\((F_k)\) has value 1. For all new time series \(F_k\) which Outlier\((F_k)\) is equal to 0, the process jumps to the step 6, otherwise it proceeds with the next step (step 5). This means that if the relation between new time series and clusters represent a change in structure, new clusters are generated (step 5). In the other case, new time series are assigned to existing clusters and then the prototypes are updated (step 6).

**Step 5:** Create new clusters. Two pieces of criteria have been defined in step 4 in order to find outlier time series. If the criteria denote that arrived time series or a group of similar arrived time series cannot be clustered properly, then one or more new clusters are created for them. Afterward, the outliers are assigned to new clusters based on their similarity and distances. In this step, before creating new clusters, outlier time series need to be classified by following this procedure. First, the pair-wise distance among new outlier time series are calculated. Next, the maximum distance between all pair-wise distances is declared. Finally, a new cluster is made from each group of outliers based on their similarity. The time series \(F_e\) and \(F_k\) are grouped if:

\[
d_{ek} < \max\{d(F_u, F_j)\}, \quad \forall F_u, F_j \in V_u, \ u \in \{1, \ldots, c\}
\]

(14)

where, if the distance of two different outlier time series is less than the maximum distance of all pairs of time series in all clusters, they are considered a group and a new cluster is generated for them.

**Step 6:** Class movement. This step includes assigning new time series to clusters and updating prototypes using the fuzzy clustering algorithm (or other distance based algorithms). To perform this step the following iterative instruction is used:

First, time series are assigned to clusters as a cluster member based on their membership values. In order to show each cluster’s member, an indicator is defined for each time series \(k\) in cluster \(i\):
\[ C_i(F_k) = \begin{cases} 1, & F_k \text{ is assigned to class } i \\ 0, & \text{else} \end{cases} \quad (15) \]

Then the prototypes of changed clusters are recalculated. Afterward, the dissimilarity matrix is updated for only changed clusters (due to alteration in a few clusters), not whole cluster structure. Subsequently, the membership matrix is also updated based on the updated distance matrix. Finally, the new memberships are utilized, in order to find whether the cluster structure of time series has changed or not. If the membership of a time series represent that its cluster needs to be changed, the iteration is continued, otherwise it proceeds to the next step. The assigning and updating of clusters periodically is summarized in Table 2.

### Table 2. Periodical part of assigning time series to existing clusters.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Periodical clustering of arrived time series</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Read centroids: (V = {V_1, \ldots, V_n}) from the (n) previous clusters (C);</td>
<td></td>
</tr>
<tr>
<td>2. For each period</td>
<td></td>
</tr>
<tr>
<td>a. Read new time series (F_k, k = {1, \ldots, m})</td>
<td></td>
</tr>
<tr>
<td>b. Compute (d_{kj} = (F_i, V_j)) /* Compute dissimilarity: */</td>
<td></td>
</tr>
<tr>
<td>c. Calculate membership degree (\mu_{kj}) of new time series.</td>
<td></td>
</tr>
<tr>
<td>d. Find outliers based on conditions 1 and 2.</td>
<td></td>
</tr>
<tr>
<td>e. If there is any outlier, classify outliers based on (d_{kj} &lt; \max{d(F_i, F_j)})</td>
<td></td>
</tr>
<tr>
<td>1) Make a new cluster for each outlier group.</td>
<td></td>
</tr>
<tr>
<td>f. Assign (F_k) to (C_j) if (d_{kj}) is minimum for (k = {1, \ldots, m}, j = {1, \ldots, n}, \text{new clusters})</td>
<td></td>
</tr>
<tr>
<td>g. Do until arriving to convergence or until next period:</td>
<td></td>
</tr>
<tr>
<td>1) Update dissimilarity matrix for changed cluster</td>
<td></td>
</tr>
<tr>
<td>2) Calculate membership of time series and Update centres (for changed clusters)</td>
<td></td>
</tr>
<tr>
<td>3) Update cluster distances</td>
<td></td>
</tr>
<tr>
<td>h. End do</td>
<td></td>
</tr>
<tr>
<td>3. End for</td>
<td></td>
</tr>
</tbody>
</table>

**Step 7:** Eliminate empty clusters. In the presented cluster structure, a cluster is omitted in two situations: (1) A cluster has to be eliminated if it did not receive new time series for a “long period”; (2) If an empty cluster keeps empty for a “short period”.

Regarding “long period”, it is defined as the number of periods for eliminating a cluster based on the number of existing clusters, arriving time series, and the average rate of accepting a time series by clusters. Pertaining to the “short period”, because in process of assigning time series to clusters and updating their centres in each cycle, moving prototypes is very slow and sometimes empty prototypes can emerge. These empty clusters need to be removed after a short period. Defining this period require more research and will be presented in a future work.

### 4. EXPERIMENTAL RESULTS

This paper, we focus on segmentation of bank customers as identifiable objects to
show how we utilize the presented method for clustering of customers. A data set is utilized which has been collected by tracking custozer’s account balances in a bank. The extracted patterns based on customer balances have been used for determining the profiles of potential customers for certain products with high correlation to their account balance. Our data set for the first experiment is a collection of time series which includes 1,750 customers and each customer is presented with his/her history of 1 year transactions with the bank. Number of transactions – the length of the time series – is different for customers. Each customer is a time series with a wide length between 150 to 350 points. From this data set, 1,000 time series of customer balances are used to make initial prototypes. Then more time series are imported periodically and added to existing clusters.

### 4.1 Results

We applied agglomerative hierarchical clustering to data set to make the initial prototypes. The prototypes of constructed clusters (10 clusters) were considered as prototypes of incremental steps. We used 50 new arrived time series for each cycle. Then $\alpha = 0.05$ is defined in order to detect time series that cause changes in the step 4. We removed a cluster when it does not receive a new time series for $T = 4$ periods, and if an empty cluster stayed without any new members for two sequential periods.

In the first cycle, 50 new time series were received. Based on mentioned conditions in the fourth step, 44 time series had the ability to be assigned to existing clusters. Other time series were defined as outlier time series or time series that represented change in the structure. In order to group these 6 time series and creating new clusters, we went to next step 5. In this case, 1 new cluster was created for 6 time series based on their similarities.

In the second cycle we faced with another 50 new time series. In this step, two new classes were created from 8 outliers that were detected earlier from 50 newly arrived time series. In Fig. 5, some of clusters are illustrated with their prototypes:

We continued the process for 15 cycles. The results of clustering are reported in Table 3. This table shows the time series added to clusters during incremental phase and also the effect of them on the cluster structure. The obvious influence is on the number of clus-
Table 3. The procedure of incremental clustering in different cycles.

<table>
<thead>
<tr>
<th></th>
<th>Initial</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
<th>C10</th>
<th>C11</th>
<th>C12</th>
<th>C13</th>
<th>C14</th>
<th>C15</th>
</tr>
</thead>
<tbody>
<tr>
<td>total objects</td>
<td>1000</td>
<td>1050</td>
<td>1100</td>
<td>1150</td>
<td>1200</td>
<td>1250</td>
<td>1300</td>
<td>1350</td>
<td>1400</td>
<td>1450</td>
<td>1500</td>
<td>1550</td>
<td>1600</td>
<td>1650</td>
<td>1700</td>
<td>1750</td>
</tr>
<tr>
<td>total classes</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>new objects</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>outliers</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>new classes</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>empty classes</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>removed classes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Squared Error</td>
<td>289</td>
<td>305</td>
<td>299</td>
<td>311</td>
<td>360</td>
<td>341</td>
<td>380</td>
<td>432</td>
<td>461</td>
<td>463</td>
<td>475</td>
<td>481</td>
<td>499</td>
<td>505</td>
<td>509</td>
<td>513</td>
</tr>
</tbody>
</table>

Fig. 6. Changes in the structure of clusters during 15 cycles.

ters which is due to creation of new clusters and omitting empty clusters. For example in fifth cycle, the algorithm has detected 3 outliers of 50 new arrived time series. Then, based on the local similarity of outliers, it has created 2 new clusters. Moreover, in cycle sixteenth, though we have a new cluster, the total of cluster numbers stay fix due to omitting a cluster.

The changes in clusters during the cycles are illustrated in Fig. 6. This diagram depicts changes in the structure of clusters during the 15 cycles. In this figure, the potential changes in different clusters are distinguished. The cluster structure change consists of increasing total number of clusters, outliers and removed clusters. Moreover, accuracy of clusters during cycles calculated by Sum of Standard Error (which is explained in section 4.3) is shown in this chart. The results indicate that the accuracy of clusters decreased slightly, however, the clusters converge after a while and the number of outliers decreases by new cycles. The intent of authors from selecting this data set is just presenting how structure of clusters change during cycles. More evaluations gained from applying the algorithm across different data sets and cardinalities are discussed in section 4.3.

4.2 Evaluation

We called our algorithm IFCMT (Incremental Fuzzy C-Mean Clustering for Time series) and compare it with different clustering algorithms in terms of execution time and accuracy. In order to show execution time and accuracy of our clustering algorithm in comparison with conventional methods, we utilized $k$-mean and hierarchal algorithms as two well-known algorithms. However, clustering of time series is an unsupervised process, same as general clustering algorithms and there are no predefined classes and no examples
that can show that the founded clusters are valid [33], therefore, it is necessary to use some validity criteria. In order to prove the merit of our approach than K-means and Hierarchy approaches for clustering time series, we run the algorithms on the bank data set. We choose different amount of records from our data set to show how it works in terms of accuracy. For initial clusters, we pick randomly 10% of data and cluster them by hierarchical clustering. Then we use the gained clusters as prototypes of k-mean and IFCMT approach to provide same conditions for clustering of data by k-mean and IFC-MT. For evaluation of clusters in terms of accuracy we utilize Sum of Squared Error (SSE), the most common measure. For each time series, the error is the distance to the nearest cluster. To get SSE, we use the following formula:

\[
SSE = \sum_{j=1}^{c} \sum_{F_i \in C_j} (dist(F_i, V_j))^2
\]

where, \( F_i \) is a time series in cluster \( C_j \) and \( V_j \) is the representative prototype for cluster \( C_j \).

The result illustrated in Fig. 7, shows the standard errors of different cardinalities of data set. It determines that dimensionally reduction of time series and incremental manner of clustering cannot lead to reduction of precision to some extend and the accuracy is competitive with other algorithms.

![Fig. 7. Accuracy of algorithms across different cardinalities.](image1)

![Fig. 8. Execution time of algorithms cross different cardinalities.](image2)

In this evaluation, the execution time is measured for the whole clustering function for convention methods and for IFCMT, the timing is calculated for all phases including data reduction, pre-clustering, making membership matrix and extending the clusters to meet all objects in 15 cycles. A DELL Server with 4GB RAM and two 2.6 GHz Quad-core Processor was used to run all tests and report the results. The results presented in Fig. 8 show timing results values (the mean accumulated CPU time in seconds), across the bank data set with different cardinalities. Despite the time required per iteration in IFCMT (very little iteration until convergence) and dimension reduction of data set, the results of this investigation demonstrate that the speed of IFCMT is better than the conventional algorithms used in literature.

Moreover, in order to investigate the effect of different data sets and demonstrate the performance of the IFCMT on them, we examined the algorithm across different branches in the mentioned bank. The averages of execution time for 5 different branches with 1,000
customers have been shown in the Fig. 9.

In another evaluation, we tried to discover the effect of selecting different initial clusters in this approach. Although the clusters are initialized by hierarchy clustering (To sum up, the reported execution timing for IFCMT algorithm was a strong implication of speedup in developed algorithm versus K-mean and Hierarchal, especially in big cordiality data sets. Not randomly), different number of initial clusters (k) has different influence on ultimate result. We have collected a dataset with 2,500 time series of customers. Then we have found the best cluster number for this data set by calculating the sum of standard error of cluster structures with different clusters (k). The best result was 28 clusters for this data set. Then we partitioned the data to initial data set (1,000 time series) and incremental data set (1,500 time series). Fig. 10 illustrates the number of clusters in each cycle during the incremental clustering process. In this graph, it is clear that the total cluster changes for a while, however after approximately 10 cycles; it remained roughly stable until the end of cycles. Moreover, this graph demonstrates that the number of initial clusters has a direct dependency on how fast the cluster structure arrives to a roughly stable state. Finally, after clustering with different initial clusters, we have calculated the standard error of each cluster structure to show how correctly selecting the number of initial clusters can affect the ultimate result. The chart in Fig. 11 demonstrates that there are not any considerable changes in the amount of standard error in terms of initial cluster numbers. However, selecting proper cluster numbers (k) in the initial data set, can lead to slightly accurate final clusters. For example in Fig. 11, (considering that the best k is 28 clusters) the lowest standard error is for the run that is around the runs whose initial clusters is near to k = 28.
5. CONCLUSIONS

The purpose of the current study was to present a method to cluster time series data efficiently. We developed a novel method for clustering time series incrementally based on its ability to accept new time series and update underlying clusters. This methodology works incrementally, and updates the structure of clusters by moving, removing, and creating clusters. Additionally, it is independent to applications and works based on user parameters according to application. The results of this study indicate that this method is much more efficient than “learning from scratch” computationally. Moreover, in terms of being accurate, this method is sufficiently accurate enough in comparison with traditional strategies. Additionally, regarding its functionality, this approach presents a viewpoint of the changes to the system clearly and helps determines the perspective of system behaviour which is of great importance for most systems.

REFERENCES


*Saeed Aghabozorgi* received his B.Sc. in Computer Engineering and Software Discipline from University of Isfahan, Iran, in 2002. He received his M.Sc. from Islamic Azad University of Najafabad, Iran, in 2005. He worked as software engineer and lecturer in Iran and Malaysia. Currently, he is a Ph.D. candidate at the University of Malaya, Kuala Lumpur, Malaysia. His current research area is data mining and web mining.

*Mahmoud Reza Saybani* received his M.Sc. from Johannes Kepler University of Linz, Austria in 1988. Currently he is a Ph.D. student at Faculty of Computer Science and Information Technology, University of Malaya, Malaysia, he worked as software engineer and Lecturer in Austria, Canada, Iran and Malaysia. His areas of interests are: data mining, fuzzy logic, mathematical modeling, natural computation, artificial immune systems, and software engineering.
**Teh Ying Wah** received his B.Sc. and M.Sc. from Oklahoma City University and Ph.D. from University of Malaya. He is currently a senior lecturer at information science department, faculty of computer science and information technology, University of Malaya. His research interests include data mining, text mining and document mining.