Incremental fuzzy clustering with multiple medoids for large data

Wang, Yangtao; Chen, Lihui; Mei, Jian-Ping

2014

Wang, Y., Chen, L., & Mei, J.-P. (2014). Incremental fuzzy clustering with multiple medoids for large data. IEEE transactions on fuzzy systems, 22(6), 1557-1568.

<https://hdl.handle.net/10356/106736>

<https://doi.org/10.1109/TFUZZ.2014.2298244>

© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/TFUZZ.2014.2298244>].

Downloaded on 26 Apr 2023 23:06:22 SGT

Incremental Fuzzy Clustering with Multiple Medoids for Large Data

Yangtao Wang, Lihui Chen, *Senior Member, IEEE*, and Jian-Ping Mei, *Member, IEEE*

Abstract—As an important technique of data analysis, clustering plays an important role in finding the underlying pattern structure embedded in the unlabelled data. Clustering algorithms that need to store the entire data into the memory for analysis become infeasible when the data set is too large to be stored. To handle such kind of large data, incremental clustering approaches are proposed. The key idea of these approaches is to find representatives (centroids or medoids) to represent each cluster in each data chunk, which is a packet of the data, and final data analysis is carried out based on those identified representatives from all the chunks. In this paper we propose a new incremental clustering approach called incremental multiple medoids based fuzzy clustering (IMMFC) to handle complex patterns that are not compact and well separated. We would like to investigate if IMMFC is a good alternative to capture the underlying data structure more accurately. IMMFC not only facilitates the selection of multiple medoids for each cluster in a data chunk, but also has the mechanism to make use of relationships among those identified medoids as side information to help the final data clustering process. The detailed problem formulation, updating rules derivation, and the in-depth analysis of the proposed IMMFC are provided. Experimental studies on several large data sets including real world malware data sets have been conducted. IMMFC outperforms existing incremental fuzzy clustering approaches in terms of clustering accuracy and robustness to the order of data. These results demonstrate the great potential of IMMFC for large data analysis.

Index Terms—Large data, incremental clustering, fuzzy clustering, multiple medoids, malware clustering.

I. INTRODUCTION

LARGE data becomes prevalent nowadays because huge amount of data can be collected easily every day from various sensors, social network sites, purchase transaction records, videos from surveillance systems to name a few. For example, 30 billion pieces of content are shared on Facebook every month, and 235 terabytes data is collected by the US Library of Congress by April 2011 [1]. According to Huber's description [2], large data is defined as data with the size of 10^8 bytes. Mining valuable information in the prevalent large data nowadays is crucial to get an edge of the competition for many parties including enterprises and government organizations. Clustering is a promising data analysis tool for finding the pattern structure and information underlining the unlabelled data. It has been widely studied in, to name a few, the areas of data mining, machine learning, information retrieval and cybersecurity. Different clustering algorithms based on various theories and applications have been developed over the past decades [3], [4], [5]. For large data clustering, one challenge is that the data is too large to be stored in the memory. Incremental clustering framework is adopted to

handle this challenge by processing data chunk by chunk. One chunk is a packet or a small portion of the entire data. The chunk size can be decided by the users. In this framework, one representative object called medoid is identified for each cluster in a chunk. However, only one medoid may not be enough to capture the underlying structure of the data. In this paper, multiple medoids are considered for representing each cluster in a chunk to improve the performance of large data clustering.

Many clustering approaches have been designed to help address large data in the literature. For example in [6], CLARA is proposed to make use of the random sampling method to handle large data. In particular, it identifies the medoids of the samples to approximate the medoids of the entire data set. Each medoid is a prototypical object for representing the cluster. In [7], Ng et al. present CLARANS and the medoids are identified by building a graph structure and searching on it. Each node in the graph contains a set of medoids that represent the data set. Randomized search is used to determine the final set of the medoids. BIRCH [8] clusters data incrementally based on the in-memory balanced tree structure and summarization techniques. Since CLARA, CLARANS and BIRCH are unsuitable for non-spherical and unbalanced data, in [9], Guha et al. propose a hierarchical clustering method named CURE, which finds multiple representative objects in each cluster and employs random sampling method to handle large data. In [10], an incremental clustering algorithm for stream data based on k-Median called LSEARCH is developed using chunk-based processing style. They process data chunk by chunk and then each cluster in one chunk is represented by one weighted centroid. The LSEARCH is applied on all the centroids to get the final set of centroids for the entire data set. In [11], spectral clustering algorithm is extended to update the clustering results incrementally to handle the large evolving data. Recently, two approaches [12], [13] based on random sampling and random Fourier maps respectively are proposed for approximating kernel k means to solve large scale kernel clustering.

The clustering methods discussed above are all based on hard clustering which indicates whether an object belongs to a certain cluster or not. Many methods have been extended to soft or fuzzy clustering approaches in order to handle more real world data sets where data objects in those data sets may not be well separated. It has been discussed that soft clustering such as those popular ones in the literature [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] may capture the natural structure of a data set more closely. Each object in a data set may belong to all clusters with various

degrees of memberships. To handle large data, an online Nonnegative Matrix Factorization approach[24] is proposed for efficiently clustering large document data set. Several incremental fuzzy clustering algorithms based on the well known Fuzzy c means(FCM) [25] have been developed. The popular algorithms include the single-pass FCM referred as SPFCM [26] and online FCM called OFCM [27], [28]. The two algorithms process data chunk by chunk and estimate the c centroids for the entire data set by extracting the information in each chunk. The kernel versions of them called spkFCM and okFCM [29] are also developed for clustering large data efficiently. The SPFCM and OFCM are all based on a vector representation of data or be referred as object data. Recently incremental fuzzy clustering for relational data have also been proposed. In [30], two algorithms based on fuzzy c medoids(FCMD)[31] called online fuzzy c medoids(OFCMD) and history based online fuzzy c medoids(HOFCMD) are developed for clustering large relational data sets. However, as discussed in [32], one medoid may not be sufficient enough to capture the underlying structure of a cluster. In order to represent the structure of a cluster more accurately, multiple medoids may be a better choice.

Inspired by the above idea, we propose a new incremental fuzzy clustering approach called incremental multiple medoids based fuzzy clustering(IMMFC) for large relational data clustering. In IMMFC, the data is processed one chunk at a time instead of loading the entire data into the memory for analysis. Multiple medoids are identified for each cluster in a data chunk by introducing a weight for each object, which measures how well the object represents the cluster in a chunk. These medoids are used as the representatives of the data set to carry out the final data partition. We also propose the mechanism to make use of some relationships of identified medoids in each chunk as side information to guide the generation of the final set of medoids. In this paper, the detailed formulation, derivation and an in-depth analysis of the model are given. The experiments of IMMFC on several large data sets including two real world malware data sets show that IMMFC achieves better clustering accuracy and robustness to the order of the data than the related chunk based algorithms. Throughout this paper, the following denotations are used unless otherwise stated: an N-dimensional data set, X , whose relational matrix is denoted by $R_{n \times n}$, and letters in bold lowercase denote vectors, e.g., $\mathbf{w}$, while bold $\mathbf{1}_n$ is a vector with length n , in which all elements are 1s.

The rest of the paper is organized as follows: in the next section, a review on the related incremental fuzzy approaches reported in the literature is highlighted. In section 3, the details of the proposed incremental clustering approach IMMFC are presented. Experiments on several large data sets are conducted and the results are analyzed in section 4. Finally, conclusions are drawn in section 5.

II. RELATED WORK

In this section, four incremental fuzzy clustering algorithms recently proposed are reviewed. The common and unique characteristics of each of the four approaches are discussed.

A. SPFCM and OFCM

SPFCM[26] and OFCM[27] are two incremental fuzzy approaches designed based on FCM. To handle large data, SPFCM and OFCM use the same strategy which is to process the data chunk by chunk. For each chunk, a set of centroids is calculated to represent the chunk with one centroid per cluster. Instead of applying FCM directly on each chunk, weighted FCM(wFCM) is used for both approaches to determine the final set of centroids.

The significant difference of SPFCM and OFCM is the way of handling the centroids of each chunk. In SPFCM, the centroids identified from the previous chunk are combined into next chunk and the final set of centroids for the entire data is generated after last chunk is processed. While in OFCM, the identification of centroids for every chunk is processed individually and an additional step is needed to generate the final set of centroids for the entire data.

In SPFCM, the weight for the centroid of each cluster in each chunk is calculated as follows:

$$w_c = \sum_{i=1}^{n_p+m} (u_{ci})w_i, \quad 1 \leq c \leq k \quad (1)$$

Where, w_c is the weight of centroid of the c_{th} cluster, n_p is the number of objects in p_{th} chunk, k is the number of clusters, u_{ci} is the membership of object i belongs to cluster c and w_i is the weight of object i . For the first chunk of the data ($p = 1$), w_i is assigned to 1 for every object and $m = 0$. From the second chunk of data ($p \neq 1$), $m = k$ and the k weighted centroids are combined with the p_{th} chunk of data. The $n_p + k$ objects will be clustered by wFCM in which the weights of the n_p objects in p_{th} chunk are all set to 1 and the weights of k objects are calculated from previous chunk. These steps continue until last chunk of data is processed.

In OFCM, the weight for the centroid of each cluster in each chunk is calculated as follows:

$$w_c = \sum_{i=1}^{n_p} (u_{ci})w_i, \quad 1 \leq c \leq k, \quad w_i = 1, \quad \forall 1 \leq i \leq n_p \quad (2)$$

The centroids are identified independently based on each chunk. The final set of centroids is generated by applying wFCM on the weighted centroids identified from all chunks.

As discussed above, both of the approaches handle large data by processing data chunk by chunk. The data type which the two approaches support is object data. However, if the input data is relational data, these approaches are not feasible. Next, we review two incremental fuzzy c medoids clustering algorithms which use similar mechanism as SPFCM and OFCM, but they are extended from handling the large object data sets to large relational data sets.

B. HOFCMD and OFCMD

In [30], two methods called HOFCMD and OFCMD are proposed for large relational data sets. Compared to SPFCM and OFCM, the major difference is the way they choose the representative of a cluster. In SPFCM and OFCM, the

representative which is called centroid is achieved by calculating the average value of the vector represented objects in a cluster. While in HOF CMD and OF CMD, the representative which is called medoid is achieved by selecting one of the real objects in the cluster. This difference makes SPFCM and OFCM always be applied to object data, and HOF CMD and OF CMD for relational data. In some applications, medoids based methods have an advantage of better interpretability by using medoids instead of centroids because medoids are real objects in the data set.

For handling large data, the incremental ways of HOF CMD and OF CMD are similar to SPFCM and OFCM respectively. Similar to wFCM, weighted fuzzy c medoids(wFCMD) is used with the objective function as follows:

$$J_{wFCMD} = \sum_{c=1}^k \sum_{i=1}^n u_{ci}^m w_i Dis(x_i, \delta_c) \quad (3)$$

Where u_{ci} is the membership of object i belongs to cluster c , k is the number of clusters, $m > 1$ is the fuzzification constant, w_i is the weight of object i , and δ_c is the medoid for cluster c . $Dis(x_i, v_c) = \|x_i - \delta_c\|$ is the distance between object x_i and medoid δ_c .

In HOF CMD, similar to SPFCM the identified weighted medoids in previous chunk are combined with current chunk as history information. Final set of medoids is generated after processing the last chunk of data. In OF CMD, similar to OFCM, the final set of medoids is generated by applying wFCMD on the medoids identified from all the chunks. The approach clusters all the chunks independently and aggregates all the medoids of the chunks in the end.

As discussed above, all the four methods use only one centroid or one medoid to represent each cluster. In our method, we identify multiple medoids to represent one cluster of each chunk and make use of some relationships of the medoids as side information to help the generation of the final medoids for the entire data set. Next, we propose our new incremental fuzzy clustering approach called IMMFC, including the detailed formulation, derivation and an in-depth analysis.

III. THE PROPOSED APPROACH

In this section, we first formulate the objective function of the proposed approach IMMFC integrating two types of mechanisms: selection of multiple medoids for each cluster and automatic collection of side information from identified medoids. The selection of multiple medoids is based on representative weight which reflects how well an object represents the cluster. The side information is collected in the form of a pair-wise constraints set. Each pair in the constraint set consists of two objects which have a high probability to be clustered in the same cluster. It is showed that the IMMFC clustering is actually an important optimization problem of the cost function with constraints. The updating rules are derived by applying the Lagrangian Multiplier method and Karush-Kuhn-Tucker(KKT) conditions. Next, we introduce the algorithm of IMMFC including the detail steps. The time complexity of the algorithm will be discussed as well.

In summary, multiple medoids are identified by IMMFC to represent each cluster in a chunk. Some relationships among those identified medoids are used to form pair-wise constraints as side information. The final set of medoids can be identified with the help of those chunk-medoids and their pair-wise constraints.

A. Objective function of IMMFC

First the following objective function is formulated for IMMFC J_{IMMFC} :

$$J_{IMMFC} = \sum_{c=1}^k \sum_{i=1}^n \sum_{j=1}^n u_{ci} v_{cj} r_{ij} + \frac{T_u}{2} \sum_{c=1}^k \sum_{i=1}^n (u_{ci})^2 + \frac{T_v}{2} \sum_{c=1}^k \sum_{j=1}^n (v_{cj})^2 - S_u \sum_{x_m, x_n \in Q} \sum_{c=1}^k u_{cm} u_{cn} \quad (4)$$

subject to

$$\sum_{c=1}^k u_{ci} = 1, \text{ for } i = 1, 2, \dots, n \quad (5)$$

$$u_{ci} \geq 0, \text{ for } c = 1, 2, \dots, k, i = 1, 2, \dots, n \quad (6)$$

$$\sum_{j=1}^n v_{cj} = 1, \text{ for } c = 1, 2, \dots, k \quad (7)$$

$$v_{cj} \geq 0, \text{ for } c = 1, 2, \dots, k, j = 1, 2, \dots, n \quad (8)$$

In J_{IMMFC} , r_{ij} is the distance or dissimilarity of object i and object j , u_{ci} is the fuzzy membership which represents the degree of object i belongs to cluster c , v_{cj} is a weight that reflects the degree of the representative of object j for cluster c . The higher the value is, the better this object can represent this cluster, and we call it representative weight in this paper.

J_{IMMFC} consists of four terms. The first term is the total dissimilarity within each cluster weighted by u_{ci} and v_{cj} . The next two terms are the regularization terms which are used as penalties on the two fuzzy memberships. $T_u > 0$ and $T_v > 0$ are two parameters to control the weight of the two regularization terms in the objective function. The fourth term is the pairwise constraint term served as the side information to supervise the final clustering. In this term, $S_u > 0$ is the parameter to tune the weight of this term in the objective function. Q denotes the set of pairs of objects in which each pair of objects has a high probability of being in the same cluster. The higher probability the objects in the pair are in the same cluster, the larger the value of term $\sum_{c=1}^k u_{cm} u_{cn}$. When object m and object n are in the same cluster, the value of $\sum_{c=1}^k u_{cm} u_{cn}$ tends to be 1, which is the maximum

value. In other words, the fourth term $\sum_{c=1}^k u_{cm} u_{cn}$ needs to be maximized in the objective function to ensure each pair of objects in set Q is clustered into the same cluster.

The clustering task is to minimize the J_{IMMFC} given set Q , and subject to the constraints in (5), (6), (7) and (8). Each

pair of constraint in set Q is constructed based on the representative weight of medoids identified from each chunk. In this work, the pairs of constraints are generated automatically after clustering each chunk. We treat the clustering process as solving the optimization problem of the cost function with constraints. By applying the Lagrangian Multiplier method and KKT conditions, we derive the membership updating rules for the new IMMFC. The Lagrangian function considering the constraints is given as follows:

$$L_{IMMFC} = J_{IMMFC} + \sum_{i=1}^n \lambda_i (\sum_{c=1}^k u_{ci} - 1) + \sum_{c=1}^k \beta_c (\sum_{j=1}^k v_{cj} - 1) + \sum_{c=1}^k \sum_{i=1}^n \psi_{ci} u_{ci} + \sum_{c=1}^k \sum_{j=1}^n \phi_{cj} v_{cj} \quad (9)$$

where the λ_i , β_c , ψ_{ci} and ϕ_{cj} are the Lagrange multipliers, and the KKT conditions for solving u_{ci} are given as follows:

$$\frac{\partial L_{IMMFC}}{\partial u_{ci}} = 0, \quad (10)$$

$$\psi_{ci} \geq 0, \quad (11)$$

$$\psi_{ci} u_{ci} = 0, \quad (12)$$

The updating rule of u_{ci} can be derived as

$$u_{ci} = \begin{cases} 0 & \text{for } c \in q^- \\ \frac{S_u}{T_u} (u_{ch} + \frac{1}{|q^+|} \sum_{f \in q^-} u_{fh} - \frac{1}{|q^+|}) \\ + \frac{1}{|q^+|} - \frac{1}{T_u} (\sum_{j=1}^n v_{cj} r_{ij} - \frac{1}{|q^+|} \sum_{f \in q^+} \sum_{j=1}^n v_{fj} r_{ij}) & \text{for } c \in q^+ \end{cases} \quad (13)$$

where u_{ch} is the membership of object h to cluster c . Object h and object i is a pair of constraint in set Q , and

$$\begin{aligned} q^- &= \{c : u_{ci} = 0\} \\ q^+ &= \{c : u_{ci} > 0\} \end{aligned} \quad (14)$$

Here, q^- is the set of clusters that object i does not belong to or has zero membership value to the clusters. q^+ is the set of clusters that object i has membership value that greater than zero. And $|q^-|$, $|q^+|$ are used to represent the size of the two sets. For determining the set q^- and q^+ , we use the same method as mentioned in [32] which is outlined as follows.

Procedure for determining q^+ and q^-

- 1 Initialize $q_0^+ = \emptyset, q_0^- = 1, 2, \dots, k, s = 0$;
 - 2 $s \leftarrow s + 1, q_s^+ = q_{s-1}^+ + \{m\}, q_s^- = q_{s-1}^- - \{m\}$,
where $m = \arg \min_{c \in q_{s-1}^-} \{\sum_{j=1}^n v_{cj} r_{ij}\}$;
 - 3 Check whether $u_{fi} > 0$ computed by (13),
where $f = \arg \max_{c \in q_s^+} \{\sum_{j=1}^n v_{cj} r_{ij}\}$.
If yes, go to step 2,
else set $q^+ = q_{s-1}^+, q^- = q_{s-1}^-$ and terminate.
-

By using the similar derivation method, the updating rule of v_{cj} is given as follows:

$$v_{cj} = \begin{cases} 0 & \text{for } j \in p^- \\ \frac{1}{|p^+|} - \frac{1}{T_v} (\sum_{i=1}^n u_{ci} r_{ij} - \frac{1}{|p^+|} \sum_{f \in p^+} \sum_{i=1}^n u_{fi} r_{ij}) & \text{for } j \in p^+ \end{cases} \quad (15)$$

where

$$\begin{aligned} p^- &= \{j : v_{cj} = 0\} \\ p^+ &= \{j : v_{cj} > 0\} \end{aligned} \quad (16)$$

Here, p^- and p^+ are the sets of objects with zero representative weight for cluster c and greater than zero representative weight for cluster c respectively. $|p^-|$ and $|p^+|$ represent the number of objects in set p^- and p^+ respectively. These two sets are determined by using the similar method as that of q^- and q^+ . The procedure for determining them is outlined as follows.

Procedure for determining p^+ and p^-

- 1 Initialize $p_0^+ = \emptyset, p_0^- = 1, 2, \dots, n, s = 0$;
 - 2 $s \leftarrow s + 1, p_s^+ = p_{s-1}^+ + \{m\}, p_s^- = p_{s-1}^- - \{m\}$,
where $m = \arg \min_{j \in p_{s-1}^-} \{\sum_{i=1}^n u_{ci} r_{ij}\}$;
 - 3 Check whether $v_{cg} > 0$ computed by Eq. 15,
where $g = \arg \max_{j \in p_s^+} \{\sum_{i=1}^n u_{ci} r_{ij}\}$.
If yes, go to step 2,
else set $p^+ = p_{s-1}^+, p^- = p_{s-1}^-$ and terminate.
-

Next, we propose the details of incremental fuzzy clustering algorithm IMMFC.

B. IMMFC Algorithm

The proposed IMMFC processes each chunk of data individually. The final set of medoids can be identified from the chunk-medoids with their pair constraints as the side information.

The main differences between IMMFC and the methods reported in the literature are (1) a new mechanism is provided to select multiple medoids instead of a single one to represent each of the clusters in each chunk, (2) both the fuzzy membership and representative weight of objects in the objective function are updated iteratively in IMMFC, (3) the auto-generated pairwise constraints from those identified medoids are used as the side information for determining the final set of medoids. The IMMFC approach is outlined as follows.

First, data set X is partitioned into non-overlapping partitions of $X = \{X_1, X_2, \dots, X_P\}$. The number of objects in each chunk X_p is n_p . As showed in the following algorithm, $R_{n_p \times n_p}^p$ is the distance matrix of chunk p which is calculated based on the data in the p_{th} chunk. In step 1, medoids set M and pairwise constraints set Q are initialized. In step 2, the fuzzy membership matrix U_p and representative weight matrix V_p for each chunk p are calculated by using Algorithm-1 with $S_u = 0$. In Algorithm-1, U_p is initialized firstly. V_p and U_p are alternatively updated by using (15) and (13) respectively

until the difference between successive estimation of U_p is small enough. v_{cj}^p which is the element of matrix V_p is the representative weight of object j for cluster c of chunk p . Then, t objects with most representative weight are identified to be medoids for cluster c of chunk p . The two medoids with top 2 representative weight are taken to form a pair of constraint automatically and added into set Q . In step 3, distance matrix $R_{|M| \times |M|}$ is calculated from medoids set M . We then calculate the membership matrix U and V by using Algorithm-1 with a suitable $S_u \neq 0$ and set $Q \neq \emptyset$. The final medoids set M^f is identified in step 4 in which the object in each cluster with the highest representative weight is selected as the medoid. In the final step, cluster indicator $\mathbf{q}$ is determined for each object. $\mathbf{q}_j$ is the cluster number which object j belongs to. This is acquired by assigning object j to the cluster whose medoid is nearest to object j .

It can be seen that some parameters need to be set before running the algorithm. It includes the stopping criterion ϵ , chunk size n_p , parameters T_u , T_v and S_u , number of clusters k , and number of medoids for every cluster t . The stopping criterion ϵ and chunk size n_p are set by user according to the data set. T_u and T_v are the parameters that control the contribution of regularization terms into the objective function. S_u controls the weight of pairwise constraints term in the objective function. For tuning the three parameters, we give the guideline to set $T_v = T_u n_p / k$ and $S_u = T_u N / (2q)$ to make the three terms have the same magnitude. Here, N is the number of medoids identified from all the chunks and q is the number of pair of constraints in set Q . Therefore, T_u can be tuned manually according to the specific data set and T_v , S_u can be calculated accordingly. About setting the number of clusters for every chunk, we applied the same principle as discussed in [27]. Suppose the number of classes of the entire data set is known as k . The same k is used as the number of clusters in each chunk. As pointed out by [27], this may lead to overclustering of each chunk because each chunk has only a small percentage data of the entire data set, and the number of classes of each chunk may be less than k . However, the overclustering unlikely causes information loss [27]. Therefore we set the number of clusters to be k -the number of classes in the data set for each chunk.

The time complexity of IMMFC is $O(P(n_p^2 + t \cdot k) + (|M|^2 + k))$ considering the chunk number P and cluster number k . $O(P(n_p^2 + t \cdot k))$ is the time complexity of processing P chunks and identifying t medoids for each cluster. $O(|M|^2 + k)$ is the time complexity of the final clustering where $|M|$ is the number of identified medoids from all the chunks. In real application, $t \cdot k$ is always much less than n_p^2 and k is much less than $|M|^2$. Therefore the time complexity can be considered as $O(P \cdot n_p^2 + |M|^2)$. To speed up the computation, we can set the elements in distance matrix with small value to 0 or using sampling methods as reported in [33]. Next, we apply IMMFC on a synthetic example to illustrate the working principle of the algorithm. It provides us some insights into IMMFC.

Algorithm: IMMFC

Input: Distance matrix of p_{th} chunk $R_{n_p \times n_p}^p$, number of clusters k , number of medoids for every cluster t , parameter T_u , T_v and S_u , stopping criterion ϵ

Output: Cluster Indicator $\mathbf{q}$

Method:

- 1 Initialize Medoids set $M = \emptyset$,
Pairwise constraints set $Q = \emptyset$.
 - 2 **for** $p = 1$ to P
 Calculate U_p, V_p using Algorithm-1
 with input parameter $S_u = 0$
 for $c = 1$ to k
 $s = 0$, set $A = 1, 2, \dots, n_p$;
 Repeat
 $s = s + 1$
 $z = \arg \max_{j \in A} v_{cj}^p$; medoid: $m_s = x_z$
 $M = M \cup \{m_s\}$;
 $A = A \setminus z$;
 Until($s=t$)
 Take the two most representative medoids for
 cluster c to form a pairwise constraint (m_1, m_2)
 and add it into set Q : $Q = Q \cup \{(m_1, m_2)\}$
 end for
 end for
 - 3 Calculate U, V using Algorithm-1 with input
parameter $S_u > 0$ based on $R_{|M| \times |M|}$ and
pairwise constraints set Q
 - 4 Final medoids set $M^f = \emptyset, c = 0$
 Repeat
 $c = c + 1$
 $z = \arg \max_{1 \leq j \leq |M|} v_{cj}$; medoid: $m_c = x_z$
 $M^f = M^f \cup \{m_c\}$;
 Until($c=k$)
5 $\mathbf{q}_j = \arg \min_{1 \leq c \leq k} Dis(m_c, x_j)$,
 $j = 1, 2, \dots, n$. $n = \sum_{p=1}^P n_p$
-

Algorithm-1:

Input: Distance matrix $R_{n \times n}$, number of clusters k ,
parameter T_u, T_v, S_u , stopping criterion ϵ

Output: Fuzzy membership matrix U ,
representative weight matrix V

Method:

- 1 Initialize U_p^0 , set iteration number $l \leftarrow 0$
 - 2 **Repeat**
 $l \leftarrow l + 1$
 Update V_p^l using (15) based on U_p^{l-1}
 Update U_p^l using (13) based on V_p^l
 Until($\|U_p^l - U_p^{l-1}\| < \epsilon$)
-

C. Synthetic example

We apply IMMFC on the 2D15 ($n=5000, k=15, d=2$) data set to show the process of the algorithm. 2D15¹ is a synthetic data set which is composed of 5000 two dimensional points with 15 classes. The distribution of the points is showed in

¹This data set was designed by Ilia Sidoroff and can be downloaded on <http://www.uef.fi/en/sipu/datasets>.

Fig. 1: 2D15 synthetic data

Fig. 1. The data is considered as coming in four chunks randomly. For each cluster we choose two medoids. The process of algorithm is showed in Fig. 2. The data distribution of the first chunk is showed in Fig. 2(a), the black pentagrams represent the medoids for each cluster. It can be seen in the figure that the two selected medoids are in the central area of each cluster. Then the same procedures are conducted for the other three chunks as showed in Fig. 2 (b), (d) and (e) respectively. Pairwise constraints set consists of all the medoids. Each two selected medoids forms one pair of constraint in the set. Then all the medoids identified from each chunk are used as the objects with the help of side information given in the set to find the final medoids for the entire data set which is showed in Fig. 2(c). It is noted that the final medoids founded are located in the ideal position in the data set. The evaluation of the clustering results will be showed in next section.

IV. EXPERIMENTAL RESULTS

In this section, experimental studies of the proposed approach are conducted on five data sets including two real world malware data sets. In the experiments, we compare the performance of IMMFC with OFCMD and HOF CMD to show the clustering results and robustness to the order of the data. For interesting application in software security, two malware data sets are used for the study. Three types of experiments are conducted and reported. First, we compare IMMFC with OFCMD and HOF CMD to see if using multiple medoids to represent each chunk improves the performance of clustering. Second, other than the two incremental fuzzy approach, we also compare IMMFC with our previous work PFC[32] which is a non-incremental fuzzy approach considering multiple medoids to see if the proposed IMMFC is better than PFC. Third, the experiments of parameter analysis are conducted with various values to show the impact of parameters. The experiments implemented in Matlab were conducted on a PC with four cores of Intel I5-2400 with 24 gigabytes of memory.

A. Data sets

We compare the performance of the algorithms on the following data sets.

2D15: The characteristic of this synthetic data is given in the above section.

MNIST²: This data set is composed of 10 classes which are 0 to 9 handwritten digit images. There are 70000 28×28 pixel images. We normalize the pixel value to [0,1] by dividing 255 and each image is represented as a 784 dimensional feature vector.

Forest³: This data set is from United States Geological Survey and United State Forest Service and it has 7 classes, 581012 objects. Each object is represented as a 54 dimensional feature vector.

Malware24: For malware data, it is referred as Malware 24 since there are 24 classes and the total number of objects is 3131. The data set is acquired from [34] whose general characteristic is showed in Table. I. The class size of the data set ranges from 26 to 300. The feature is extracted by Malheur based on the MIST representation[34] of malware behavior.

Malware28: A larger malware data set which we refer as Malware 28 is created based on the above malware data by adding 4 new malware classes. The detail of this data set is showed in Table. II. There are 28 malware classes with class size ranging from 26 to 1575 and the total number of objects is 6140.

TABLE I: Malware24 data set

Malware class	class#	Malware class	class#
Adultbrowser	262	Porndialer	97
Allapple	300	Rbot	101
Bancos	48	Rotator	300
Casino	140	Salinity	84
Dorfd	65	Spygames	139
Ejik	168	Swizzor	78
Flystudio	33	Vapsup	45
Ldpinch	43	Vikingdll	158
Looper	209	Vikingdz	68
Magiccasin	174	Virut	202
Podnuha	300	Woikoiner	50
Posion	26	Zhelatin	41

TABLE II: Malware28 data set

Malware class	class#	Malware class	class#
Adultbrowser	262	Posion	26
Allapple	300	Porndialer	97
AutoIt	1159	Rbot	101
AutoRun	103	Rotator	300
Bancos	48	Salinity	84
Basun	1575	Spygames	139
Casino	140	Swizzor	78
Dorfd	65	Udr	172
Ejik	168	Vapsup	45
Flystudio	33	Vikingdll	158
Ldpinch	43	Vikingdz	68
Looper	209	Virut	202
Magiccasin	174	Woikoiner	50
Podnuha	300	Zhelatin	41

²This data set can be downloaded on <http://yann.lecun.com/exdb/mnist/>.

³This data set can be downloaded on <http://uisacad2.uis.edu/dstar/data/clusteringdata.html>.

Fig. 2: (a), (b), (d), (e) represent the data in each chunk and its representative medoids for each cluster in the chunk, (c) represents the final medoids of the whole data set, all the medoids are marked as pentagram

B. Evaluation criterion

Three popular external criteria *Accuracy* [17], *F-measure* [35], and *Normalized Mutual Information*(NMI) [36] are used to evaluate the clustering results, which measure the agreement of the clustering results produced by an algorithm and the ground truth. If we refer *class* as the ground truth, and *cluster* as the results of a clustering algorithm, the NMI is calculated as follows:

$$NMI = \frac{\sum_{c=1}^k \sum_{p=1}^m n_c^p \log\left(\frac{n \cdot n_c^p}{n_c \cdot n_p}\right)}{\sqrt{\left(\sum_{c=1}^k n_c \log\left(\frac{n_c}{n}\right)\right) \left(\sum_{p=1}^m n_p \log\left(\frac{n_p}{n}\right)\right)}} \quad (17)$$

where n is the total number of objects, n_c and n_p are the numbers of objects in the c_{th} cluster and the p_{th} class, respectively, and n_c^p is the number of common objects in class p and cluster c . For F-measure, the calculation based on precision and recall is as follows:

$$F - measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (18)$$

where,

$$precision = \frac{n_c^p}{n_c} \quad (19)$$

$$recall = \frac{n_c^p}{n_p} \quad (20)$$

Accuracy is calculated as follows after obtaining a one-to-one match between *clusters* and *classes*:

$$Accuracy = \sum_{c=1}^k \frac{n_c^j}{n} \quad (21)$$

where n_c^j is the number of common objects in the c_{th} cluster and its matched class j . The higher the values of the three criteria are, the better the clustering result is. The value is equal to 1 only when the clustering result is same as the ground truth.

C. Initialization

For OFCMD and HOF CMD, we initialize the medoids by using the method adopted in [31]. The object which has the minimum distance to all the other objects is selected as the first medoid. The remaining medoids are chosen consecutively by selecting the objects that maximize their minimal distance with existing medoids. This helps the medoids distribute evenly in the data space to avoid converging to a bad local optimum. For the proposed IMMFC, we initialize the input fuzzy membership U by using similar method. First, we choose the medoids with the same method mentioned above. Then, if the medoid for cluster c is object i , we set $u_{ci} = 1$, and set the membership of the other objects to the same cluster as 0. The detail steps of initialization of IMMFC are as follows.

D. Results on 2D15, MNIST and Forest data sets

The parameter T_u is set to be 0.03, 0.1, 0.1 for 2D15, MNIST, and Forest data set respectively, and T_v, S_u are calculated according to the guideline. We set number of medoids for each cluster $t = 2$, and $\epsilon = 10^{-5}$ for IMMFC. We set fuzzifier $m = 1.7$ which is recommended in [29] for OFCMD and HOF CMD to make them work well for these data sets. Number of clusters k is set to be 2, 10, 7 for 2D15, MNIST and Forest data sets respectively. The entire data set is partitioned randomly into equal sized chunks. The size of each chunk is

Initialization for IMMFC

Set the number of clusters k ,
fuzzy membership matrix is $U_{k \times n}$

Calculate the first medoid:

$$p = \arg \min_{1 \leq j \leq n} \sum_{i=1}^n Dis(x_j, x_i);$$

first medoid: $v_1 = x_p$

Medoids set $V = \{v_1\}, m = 1;$

Fuzzy membership $u_{mp} = 1$, and $u_{mj} = 0$,
with $j = 1, 2 \dots n, j \neq p$

Repeat

$m = m + 1$

$p = \arg \max_{1 \leq i \leq n; x_i \notin V} \min_{1 \leq k \leq |V|} Dis(v_k, x_i);$

medoid: $v_m = x_p$

$V = V \cup \{v_m\};$

$u_{mp} = 1$, and $u_{mj} = 0$, with $j = 1, 2 \dots n, j \neq p$

Until($m=k$)

decided by the user. Normally it refers to a certain percentage of the entire data set size. The size of the last chunk maybe smaller than the others if the entire data set can not be divided by the chunk size. For 2D15 and MNIST data sets, we conduct experiments with chunk sizes as 1%, 2.5%, 5%, 10% and 25% of the entire data set size. For Forest data set, limited by memory, smaller percentages are chosen for chunk sizes as 0.1%, 0.25%, 0.5%, 1% and 2.5% of the entire data set size. To see the robustness of each approach, each of them runs 20 trials with random order of the input data. We calculate the mean, standard deviation, minimum and maximum of the values of accuracy, F-measure and NMI over 20 trials. The mean value reflects the average performance of the clustering algorithm. The standard deviation, minimum and maximum values reflect the robustness of the clustering algorithm to the order of the data. The results of the three data sets are showed in Table. III, Table. IV and Table. V respectively. From the tables we can see, IMMFC always produces the best partition every time with various chunk sizes. The results also show that IMMFC always has the lowest standard deviation over 20 trials which reflects that IMMFC is more robust to the order of the data sequence.

Table. III (a), (b), (c) are the accuracy results on 2D15, MNIST and Forest data set respectively. The results show that HOF CMD performs better than OF CMD when the chunk size is small, and OF CMD performs better than HOF CMD when the chunk size increases to a certain percentage. IMMFC performs better not only on the accuracy, but also on the robustness to the order of data and different chunk sizes. The accuracy results on MNIST and Forest data sets provide similar pattern in performance although the accuracy value becomes lower than the value on 2D15. This phenomenon was also observed in related work [29] by using the same data set. Compared with OF CMD on the three data sets, the improvements of IMMFC on average accuracy over all the chunk sizes are 3.2%, 15% and 4.3%. Compared with HOF CMD, the improvements of IMMFC are 3.9%, 57.8% and 16.1%. The results of F-measure and NMI given in Table. IV and Table. V show similar pattern as accuracy.

TABLE III: Accuracy of OF CMD, HOF CMD and IMMFC

(a) 2D15						
Chunk size	Algorithm					
	OF CMD		HOF CMD		IMMFC	
	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.9318 0.8910	0.0432 0.9934	0.9768 0.9002	0.0338 0.9942	0.9810 0.9122	0.0252 0.9936
2.5%	0.9303 0.9060	0.0340 0.9938	0.9841 0.9138	0.0248 0.9938	0.9933 0.9930	0.0002 0.9936
5%	0.9741 0.9168	0.0319 0.9936	0.9554 0.8960	0.0391 0.9928	0.9932 0.9926	0.0003 0.9938
10%	0.9792 0.9196	0.0301 0.9938	0.9253 0.9008	0.0323 0.9918	0.9932 0.9924	0.0003 0.9936
25%	0.9854 0.9230	0.0220 0.9938	0.9270 0.9028	0.0301 0.9884	0.9924 0.9904	0.0012 0.9936

(b) MNIST						
Chunk size	Algorithm					
	OF CMD		HOF CMD		IMMFC	
	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.3334 0.2919	0.0365 0.4182	0.2954 0.2129	0.0564 0.3916	0.4448 0.4106	0.0322 0.5009
2.5%	0.4069 0.3157	0.0556 0.4758	0.2687 0.2140	0.0309 0.3075	0.4352 0.4121	0.0260 0.4740
5%	0.4123 0.3150	0.0498 0.4913	0.2930 0.1381	0.0711 0.3687	0.4454 0.4091	0.0311 0.4996
10%	0.4050 0.3304	0.0357 0.4709	0.2789 0.1957	0.0510 0.3553	0.4642 0.4081	0.0289 0.5211
25%	0.4190 0.3490	0.0331 0.4657	0.3048 0.2648	0.0304 0.3720	0.4841 0.4452	0.0288 0.5362

(c) Forest						
Chunk size	Algorithm					
	OF CMD		HOF CMD		IMMFC	
	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
0.1%	0.3570 0.3300	0.0241 0.3984	0.3181 0.2724	0.0340 0.3680	0.3873 0.3633	0.0154 0.4060
0.25%	0.3836 0.3429	0.0248 0.4100	0.3348 0.2698	0.0452 0.3847	0.4058 0.3727	0.0185 0.4157
0.5%	0.3760 0.2734	0.0563 0.4053	0.3192 0.2706	0.0364 0.3893	0.3941 0.3611	0.0248 0.4145
1%	0.3830 0.3748	0.0171 0.3925	0.3410 0.2742	0.0364 0.3864	0.3959 0.3567	0.0224 0.4091
2.5%	0.3727 0.3276	0.0172 0.3935	0.3299 0.2802	0.0322 0.3766	0.3838 0.3590	0.0213 0.4090

E. Results on Malware24

For IMMFC, we set the parameters $T_u = 0.1$, number of clusters $k = 24$, number of medoids for each cluster $t = 2$, and $\epsilon = 10^{-5}$. T_v and S_u are set according to the guideline. And we set fuzzifier $m = 1.7$, and number of clusters $k = 24$ for OF CMD and HOF CMD. Each approach is run for 20 trials. The chunk sizes of all the approaches are selected as 1%, 2.5%, 5%, 10%, 25% of the entire data set. The best results of each approach with respect to different chunk sizes are tabulated in Table. VI. It can be seen that IMMFC yields higher accuracy, NMI and F-measure values than OF CMD and HOF CMD. The results show that the mechanism using multiple medoids to represent each chunk and using side information in the final labeling is more effective than the one using single medoid. Also from the standard deviation value in Table. VI, we can see that IMMFC is more robust

TABLE VI: Comparison of OFCMD, HOF CMD and IMMFC on Malware24

	Chunk Size	Accuracy	NMI	F-measure
OFCMD	1%	0.7703 $\pm$ 0.0237	0.8987 $\pm$ 0.0099	0.7865 $\pm$ 0.0208
HOF CMD	25%	0.7537 $\pm$ 0.0358	0.8946 $\pm$ 0.0121	0.7782 $\pm$ 0.0283
IMMFC	25%	0.8586 $\pm$ 0.0211	0.9251 $\pm$ 0.0077	0.8642 $\pm$ 0.0195

TABLE IV: F-measure of OFCMD, HOF CMD and IMMFC

(a) 2D15

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.9477	0.0324	0.9800	0.0279	0.9860	0.0152
	0.9087	0.9934	0.9117	0.9942	0.9406	0.9936
2.5%	0.9429	0.0274	0.9861	0.0183	0.9933	0.0002
	0.9215	0.9938	0.9347	0.9938	0.9930	0.9936
5%	0.9774	0.0264	0.9599	0.0347	0.9932	0.0003
	0.9277	0.9936	0.9020	0.9928	0.9926	0.9938
10%	0.9822	0.0239	0.9418	0.0211	0.9932	0.0003
	0.9340	0.9938	0.9192	0.9918	0.9924	0.9936
25%	0.9877	0.0148	0.9452	0.0211	0.9924	0.0012
	0.9457	0.9938	0.9198	0.9884	0.9904	0.9936

(b) MNIST

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.3611	0.0277	0.3295	0.0435	0.4799	0.0347
	0.3241	0.4180	0.2618	0.3951	0.4406	0.5176
2.5%	0.4162	0.0481	0.3052	0.0259	0.4576	0.0229
	0.3365	0.4766	0.2597	0.3414	0.4231	0.5024
5%	0.4234	0.0459	0.3210	0.0602	0.4887	0.0393
	0.3396	0.5019	0.1990	0.3869	0.4360	0.5290
10%	0.4082	0.0392	0.3122	0.0439	0.4950	0.0370
	0.3260	0.4801	0.2403	0.3823	0.4167	0.5436
25%	0.4254	0.0330	0.3401	0.0231	0.5253	0.0241
	0.3060	0.3812	0.3602	0.4711	0.4917	0.5505

(c) Forest

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
0.1%	0.4104	0.0320	0.4038	0.0418	0.4310	0.0263
	0.3827	0.4479	0.3453	0.4667	0.3917	0.4613
0.25%	0.4335	0.0207	0.4201	0.0579	0.4566	0.0009
	0.3917	0.4481	0.3349	0.4726	0.4554	0.4575
0.5%	0.4254	0.0450	0.4007	0.0363	0.4568	0.0017
	0.3398	0.4542	0.3644	0.4742	0.4554	0.4588
1%	0.4414	0.0100	0.4319	0.0422	0.4564	0.0008
	0.4253	0.4543	0.3648	0.4750	0.4552	0.4571
2.5%	0.4391	0.0184	0.4182	0.0442	0.4557	0.0022
	0.3969	0.4497	0.3653	0.4782	0.4532	0.4582

TABLE V: NMI of OFCMD, HOF CMD and IMMFC

(a) 2D15

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.9609	0.0183	0.9783	0.0153	0.9850	0.0118
	0.9321	0.9866	0.9387	0.9874	0.9683	0.9865
2.5%	0.9564	0.0163	0.9798	0.0130	0.9856	0.0004
	0.9432	0.9866	0.9442	0.9869	0.9851	0.9863
5%	0.9772	0.0149	0.9656	0.0187	0.9854	0.0007
	0.9463	0.9863	0.9379	0.9848	0.9842	0.9865
10%	0.9794	0.0143	0.9552	0.0126	0.9854	0.0006
	0.9483	0.9866	0.9432	0.9834	0.9841	0.9862
25%	0.9820	0.0075	0.9584	0.0103	0.9839	0.0020
	0.9610	0.9864	0.9428	0.9791	0.9807	0.9862

(b) MNIST

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
1%	0.2642	0.0287	0.2087	0.0461	0.3970	0.0231
	0.2218	0.3323	0.1321	0.2767	0.3704	0.4290
2.5%	0.3044	0.0334	0.1906	0.0310	0.3957	0.0298
	0.2494	0.3493	0.1286	0.2348	0.3637	0.4140
5%	0.3019	0.0313	0.2006	0.0698	0.4035	0.0263
	0.2533	0.3473	0.0500	0.2830	0.3681	0.4367
10%	0.2868	0.0343	0.1986	0.0439	0.3993	0.0381
	0.2228	0.3564	0.1315	0.2710	0.3351	0.4343
25%	0.2961	0.0277	0.2230	0.0275	0.4161	0.0187
	0.2403	0.3416	0.1873	0.2668	0.3976	0.4477

(c) Forest

	Algorithm					
	OFCMD		HOF CMD		IMMFC	
Chunk size	avg. min.	std. max.	avg. min.	std. max.	avg. min.	std. max.
0.1%	0.1383	0.0060	0.1329	0.0098	0.1521	0.0045
	0.1389	0.1480	0.0746	0.1562	0.1461	0.1574
0.25%	0.1440	0.0064	0.1432	0.0085	0.1560	0.0046
	0.1453	0.1530	0.1306	0.1558	0.1479	0.1597
0.5%	0.1462	0.0087	0.1385	0.0117	0.1581	0.0074
	0.1351	0.1562	0.1260	0.1601	0.1492	0.1685
1%	0.1525	0.0044	0.1376	0.0088	0.1562	0.0025
	0.1557	0.1610	0.1275	0.1523	0.1527	0.1581
2.5%	0.1529	0.0071	0.1432	0.0123	0.1601	0.0106
	0.1468	0.1610	0.1187	0.1575	0.1455	0.1723

compared to OFCMD and HOF CMD with respect to the order of the data.

F. Results on Malware28

For Malware28 data set, we set the parameters for each algorithm the same as the setting in the above section. Table. VII shows best results of accuracy, NMI and F-measure of OFCMD, HOF CMD and IMMFC with respect to different chunk sizes. In each trial, the order of data is randomized. From the mean value and standard deviation of the three

criteria, it can be seen that IMMFC still performs better than OFCMD and HOF CMD on the larger malware data set with more accurate clustering performance and more robust to the order of the data.

G. Compare with PFC

We also compared the IMMFC with our previous work PFC, which is the only existing non-incremental and multiple medoids fuzzy clustering approach for relational data in the literature. We wish to see if the incremental approach causes

TABLE VII: Comparison of OFCMD, HOF CMD and IMMFC on Malware28

	Chunk Size	Accuracy	NMI	F-measure
OFCMD	1%	0.8007 $\pm$ 0.0603	0.8818 $\pm$ 0.0225	0.7934 $\pm$ 0.0418
HOF CMD	25%	0.8040 $\pm$ 0.0380	0.8983 $\pm$ 0.0140	0.8053 $\pm$ 0.0300
IMMFC	5%	0.9107 $\pm$ 0.0267	0.9431 $\pm$ 0.0143	0.9060 $\pm$ 0.0290

any obvious trade off in terms of clustering accuracy when compared with the non-incremental approach using entire data set.

For both IMMFC and PFC, the simulations are run under the same parameter setting and using the same initialization method as in the above section. Since PFC is non-incremental approach which needs to store the entire distance matrix in the memory, we are only able to get PFC results on 4 out of 5 data sets with our current computer configuration (RAM: 24G), limited by the memory size. IMMFC is run for 20 trials and the mean values of the three evaluation criterions are used to compare with PFC. Because the result of every trial of PFC is the same when the initialization method is used, PFC only needs to be run once. The best results of both approaches are tabulated in Table. VIII, where the chunk size of IMMFC varies as indicated in the table. From the table we find that IMMFC outperforms PFC on 2D15, Malware24, Malware28 and has comparable results with PFC on MNIST. These results show that IMMFC gets even higher or comparable clustering accuracy than the non-incremental and entire data set based approach.

H. Parameter Analysis

To show both the effectiveness of the guideline and the impact of the parameters on the performance of IMMFC, we set T_u in different values as: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3 and 1. T_v and S_u are calculated according to the guideline discussed in section III B. The following Fig. 3 shows the accuracy results of IMMFC on the 2D15, MNIST and Forest with chunk sizes 5%, 5%, 0.5% respectively. Here we only show the accuracy results, the results of F-measure and NMI have the similar pattern. From Fig. 3 we notice that the parameter T_u and its corresponding T_v and S_u can significantly affect the performance of IMMFC. As given in Fig. 3 (a), (b), (c), when T_u is set to be 0.03, 0.1 and 0.1 respectively, IMMFC achieves the best performance. Generally, very small or very large value of T_u will lead to degenerated performance of IMMFC. Therefore, we recommend to tune T_u between 0.01 and 0.3 in practice.

V. CONCLUSION

We have proposed a new incremental fuzzy clustering approach called IMMFC for large data analysis, and apply IMMFC on large image data and malware data to demonstrate its feasibility and potential. IMMFC processes large data chunk by chunk. One distinctive characteristic of IMMFC is that the mechanism is provided to capture and represent each of the clusters in a data chunk by multiple medoids not a single medoid. The other characteristic is that some pairwise constraints are automatically generated from the medoids

identified throughout every chunk and used to help the final data partition. Experiments conducted on a synthetic data set, and several large data sets including two real world malware data sets show IMMFC outperforms related incremental algorithms with more accurate clustering results and more robust to the order of data. Moreover, IMMFC achieves higher or comparable clustering results compared with the related non-incremental algorithm. The merits showed in the experiments indicate that IMMFC has a great potential to be used for large data clustering. In the future, much larger data sets need to be collected and tested. Further experimental studies may be conducted to investigate the most appropriate number of medoids for a specific data set.

APPENDIX A

DERIVATION OF UPDATING RULES

The detail derivation of fuzzy membership matrix U and representative matrix V are given here based on the Lagrangian function in (9) and KKT conditions in (10), (11), and (12). Derived by (10) and (5), u_{ci} is given as:

$$u_{ci} = \frac{S_u}{T_u} \left(u_{ch} - \frac{1}{k} \right) + \frac{1}{k} - \frac{1}{T_u} [(\psi_{ci} + \alpha_{ci}) - \frac{1}{k} \sum_{f=1}^k (\psi_{fi} + \alpha_{fi})], \quad (22)$$

where

$$\alpha_{ci} = \sum_{j=1}^n v_{cj} r_{ij}, \quad (23)$$

We consider the condition in (11) from two cases respectively.

- 1) $\psi_{ci} = 0, \forall c \in 1, 2 \dots k$, then (22) becomes

$$u_{ci} = \frac{S_u}{T_u} \left(u_{ch} - \frac{1}{k} \right) + \frac{1}{k} - \frac{1}{T_u} [\alpha_{ci} - \frac{1}{k} \sum_{f=1}^k \alpha_{fi}], \quad (24)$$

For every object, this set of solution is only valid if $\forall c \in 1, 2 \dots k$

$$\frac{S_u}{T_u} \left(u_{ch} - \frac{1}{k} \right) + \frac{1}{k} - \frac{1}{T_u} [\alpha_{ci} - \frac{1}{k} \sum_{f=1}^k \alpha_{fi}] \geq 0, \quad (25)$$

otherwise, the next case need to be considered.

- 2) $\psi_{ci} \geq 0$ for at least on c . According to KKT condition (12), when $\psi_{ci} > 0$, $u_{ci} = 0$, and when $u_{ci} > 0$, $\psi_{ci} = 0$. Therefore, we split the set of clusters to two subsets named q^- and q^+ according to constraint in (6), with

$$\begin{aligned} q^- &= \{c : u_{ci} = 0\} \\ q^+ &= \{c : u_{ci} > 0\} \end{aligned} \quad (26)$$

so for $c \in q^-, u_{ci} = 0$, and we need derive u_{ci} when $c \in q^+$. Because $c \in q^+, u_{ci} > 0$, then $\psi_{ci} = 0$, and

TABLE VIII: Comparison of IMMFC and PFC

Data Set	Algorithm	Chunk Size	Accuracy	NMI	F-measure
2D15	PFC	100%	0.9166	0.9587	0.9406
	IMMFC	2.5%	0.9933	0.9856	0.9932
MNIST	PFC	100%	0.5192	0.4358	0.5406
	IMMFC	25%	0.4841	0.4161	0.5253
Forest	PFC	100%	—	—	—
	IMMFC	0.25%	0.4058	0.1560	0.4566
Malware24	PFC	100%	0.7352	0.8805	0.7545
	IMMFC	10%	0.8524	0.9221	0.8582
Malware28	PFC	100%	0.8109	0.8919	0.8114
	IMMFC	10%	0.8668	0.9245	0.8562

Fig. 3: (a), (b), (c) show accuracy of IMMFC on 2D15, MNIST and Forest with various value of parameters

(22) is rewritten as:

$$u_{ci} = \frac{S_u}{T_u} (u_{ch} - \frac{1}{k}) + \frac{1}{k} - \frac{1}{T_u} \alpha_{ci} + \frac{1}{T_u} \frac{1}{k} \left[\sum_{f \in q^+} \alpha_{fi} + \sum_{f \in q^-} (\alpha_{fi} + \psi_{fi}) \right], \quad (27)$$

for $f \in q^-$, $u_{ci} = 0$, according to (22), we can get:

$$\alpha_{fi} + \psi_{fi} = S_u (u_{ch} - \frac{1}{k}) + \frac{T_u}{k} + \frac{1}{k} \sum_{m=1}^k (\alpha_{mi} + \psi_{mi}) \quad (28)$$

Then, after several steps of derivation, the following equation is obtained:

$$\sum_{f \in q^-} (\alpha_{fi} + \psi_{fi}) = \frac{k S_u}{|q^+|} \sum_{f \in q^-} u_{fh} - \frac{S_u |q^-|}{|q^+|} + \frac{|q^-| T_u}{|q^+|} + \frac{|q^-|}{|q^+|} \sum_{f \in q^+} \alpha_{fi} \quad (29)$$

Substituting (29) to (27), we finally obtain the update equation for u_{ci} as showed in (13).

The updating equation of V as in (15) can also be derived using similar procedure of derivation.

REFERENCES

- [1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," *McKinsey Global Institute*, pp. 1–137, 2011.
- [2] P. Huber, *Massive Data Sets*. National Academy Press, 1997, ch. Massive Data Sets Workshop: The Morning After, pp. 169–184.
- [3] A. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
- [4] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern recognition*, vol. 41, no. 1, pp. 176–190, 2008.
- [5] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.
- [6] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. Wiley-Interscience, 2009, vol. 344.
- [7] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 144–155. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645920.672827>
- [8] T. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM SIGMOD international Conference on Management of Data*, vol. 1, 1996, pp. 103–114.
- [9] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.
- [10] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 3, pp. 515–528, 2003.
- [11] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, "Incremental spectral clustering by efficiently updating the eigen-system," *Pattern Recognition*, vol. 43, no. 1, pp. 113–127, 2010.
- [12] R. Chitta, R. Jin, T. Havens, and A. Jain, "Approximate kernel k-means: Solution to large scale kernel clustering," in *Proc. ACM SIGKDD*, 2011, pp. 551–556.
- [13] R. Chitta, R. Jin, and A. K. Jain, "Efficient kernel clustering using random fourier features," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 161–170.
- [14] E. G. Mansoori, "Frbc: A fuzzy rule-based clustering algorithm," *Fuzzy Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 960–971, 2011.
- [15] B. Hartmann, O. Banfer, O. Nelles, A. Sodja, L. Teslic, and I. Skrljanc, "Supervised hierarchical clustering in fuzzy model identification," *Fuzzy Systems, IEEE Transactions on*, vol. 19, no. 6, pp. 1163–1176, 2011.
- [16] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Multiple kernel fuzzy clustering," *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 120–134, 2012.
- [17] J.-P. Mei and L. Chen, "A fuzzy approach for multitype relational data clustering," *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 358–371, 2012.
- [18] L. F. Coletta, L. Vendramin, E. R. Hruschka, R. J. Campello, and W. Pedrycz, "Collaborative fuzzy clustering algorithms: Some refine-

ments and design guidelines,” *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 444–462, 2012.

- [19] S. P. Chatzis and T. A. Varvarigou, “A fuzzy clustering approach toward hidden markov random field models for enhanced spatially constrained image segmentation,” *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 1351–1361, 2008.
- [20] S. Chatzis and T. Varvarigou, “Factor analysis latent subspace modeling and robust fuzzy clustering using formula formulatype,” *Fuzzy Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 505–517, 2009.
- [21] S. Krinidis and V. Chatzis, “A robust fuzzy local information c-means clustering algorithm,” *Image Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1328–1337, 2010.
- [22] Y.-Y. Lin, J.-Y. Chang, N. R. Pal, and C.-T. Lin, “A mutually recurrent interval type-2 fuzzy neural network (mrit2fnn) with self-evolving structure and parameters,” *Fuzzy Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 766–775, 2013.
- [23] D. Anderson, A. Zare, and S. Price, “Comparing fuzzy, probabilistic and possibilistic partitions using the earth mover’s distance,” *Fuzzy Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 492–509, 2013.
- [24] F. Wang, C. Tan, A. C. König, and P. Li, “Efficient document clustering via online nonnegative matrix factorizations,” in *Eleventh SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*, 2011.
- [25] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.
- [26] P. Hore, L. Hall, and D. Goldgof, “Single pass fuzzy c means,” in *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*. IEEE, 2007, pp. 1–7.
- [27] P. Hore, L. Hall, D. Goldgof, and W. Cheng, “Online fuzzy c means,” in *Fuzzy Information Processing Society, 2008. NAFIPS 2008. Annual Meeting of the North American*. IEEE, 2008, pp. 1–5.
- [28] P. Hore, L. Hall, D. Goldgof, Y. Gu, A. Maudsley, and A. Darkazanli, “A scalable framework for segmenting magnetic resonance images,” *Journal of signal processing systems*, vol. 54, no. 1, pp. 183–203, 2009.
- [29] T. Havens, J. Bezdek, C. Leckie, L. Hall, and M. Palaniswami, “Fuzzy c-means for very large data,” *Fuzzy Systems, IEEE Transactions on*, no. 99, p. 1, 2012.
- [30] N. Labroche, “New incremental fuzzy c medoids clustering algorithms,” in *Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American*. IEEE, 2010, pp. 1–6.
- [31] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, “Low-complexity fuzzy relational clustering algorithms for web mining,” *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 595–607, 2001.
- [32] J. Mei and L. Chen, “Fuzzy clustering with weighted medoids for relational data,” *Pattern Recognition*, vol. 43, no. 5, pp. 1964–1974, 2010.
- [33] J. Bezdek, R. Hathaway, J. Huband, C. Leckie, and R. Kotagiri, “Approximate clustering in very large relational data,” *International journal of intelligent systems*, vol. 21, no. 8, pp. 817–841, 2006.
- [34] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic analysis of malware behavior using machine learning,” *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.
- [35] B. Larsen and C. Aone, “Fast and effective text mining using linear-time document clustering,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 16–22.
- [36] A. Strehl and J. Ghosh, “Cluster ensembles—a knowledge reuse framework for combining multiple partitions,” *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

and a member of the IEEE Computational Intelligence Society.

LiHui CHEN received the BEng in Computer Science & Engineering at Zhejiang University, China and the PhD in Computational Science at University of St. Andrews, UK. Currently she is an Associate Professor in the Division of Information Engineering at Nanyang Technological University in Singapore. Her research interests include machine learning algorithms and applications, data mining and web intelligence. She has published over ninety referred papers in international journals and conferences in these areas. She is a senior member of the IEEE,

2014, she has been with the College of Computer Science and Technology, Zhejiang University of Technology, China. Her general research interests lie in the area of machine learning algorithms and applications to Web mining, text analysis, and bioinformatics.

Jian-Ping Mei received the B. Eng degree from the Ningbo University, China, in 2005, the M. Eng degree from the Zhejiang University, China, in 2007, and the PhD degree from the Nanyang Technological University, Singapore, in 2012, all in electrical and electronic engineering. She was working at the Nanyang Technological University, Singapore, from 2011 to 2013, first as a project officer in the Center of Bioinformatics Research and then as a research fellow in the Joint NTU-UBC research center of excellence in active living for the elderly. Since

Yangtao Wang received the B.Eng. and the M.Eng. degree in Mechanical and Electronic Engineering from Harbin University of Science and Technology, Harbin, China, in 2005 and 2008. He is currently working toward the Ph.D. degree with Nanyang Technological University in School of Electrical and Electronic Engineering, Singapore. His research interests include machine learning algorithms for large data analysis and applications on malware analysis.