Incremental fuzzy clustering for document categorization

Jian-Ping Mei, Yangtao Wang, Lihui Chen, and Chunyan Miao

***Abstract*— Incremental clustering has been proposed to handle large datasets which can not fit into memory entirely. Single pass fuzzy c-means (SpFCM) and Online fuzzy c-means (OFCM) are two representative incremental fuzzy clustering methods. Both of them extend the scalability of fuzzy c-means (FCM) by processing the dataset chunk by chunk. However, due to the data sparsity and high-dimensionality, SpFCM and OFCM fail to produce reasonable results for document data. In this study, we work on clustering approaches that take care of both the large-scale and high-dimensionality issues. Specifically, we propose two methods for incrementally clustering of document data. The first method is a modification of the existing FCM-based incremental clustering with a step to normalize the centroids in each iteration, while the other method is incremental clustering, i.e., *Single-Pass* or *Online*, with weighted fuzzy co-clustering. We use several benchmark document datasets for experimental study. The experimental results show that the proposed approaches achieved significant improvements over existing SpFCM and OFCM in document clustering.**

I. Introduction

GIVEN a set of objects, clustering is a process that automatically generates groups of objects called clusters so that objects in the same cluster share more common properties than those in different clusters. Clustering is an important data analysis tool that has been studied for several decades and it has become one of the most widely used knowledge discover techniques in many data mining applications that emerged recently. Extensive studies have been made to develop different clustering approaches typically with emphasis on improving one or more specific aspects of clustering including effectiveness, efficiency, robustness and scalability. Effectiveness has been the main concern in many studies, which aim to produce high qualities of clusters while scalability has become another critical issue more recently as the size of data becomes larger and larger. For a particular clustering algorithm, its performance may vary in one application from another depending on the nature of the data to be processed. In this study, we focus on the task of clustering of large document data for automatic document categorization.

Jian-Ping Mei is with the College of Computer Science and Technology, Zhejiang University of Technology, 310023 Hangzhou, China (email: meijianping10@gmail.com).

Yangtao Wang and Lihui Chen are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (email: wang0689@e.ntu.edu.sg, elhchen@ntu.edu.sg).

Chunyan Miao is with the School of Computer Engineering, Nanyang Technological University, Singapore (email: ascymiao@ntu.edu.sg).

This research is supported in part by Interactive and Digital Media Programme Office (IDMPO), National Research Foundation (NRF) hosted at Media Development Authority (MDA) under Grant No.: MDA/IDM/2012/8/8-2 VOL 01.

Document categorization is an important component in many Web mining and text mining tasks and clustering is an intuitive technique for automatic document categorization. The vector space model has been widely used to represent documents. With this model, each document is represented as a vector where each distinctive word is a feature. Since the dictionary of words is large, e.g., over several thousands, and each document only contains a small subset of words from the whole dictionary, document data are sparse and high dimensional. This makes many clustering approaches no more suitable for dealing with document data. Efforts have been made by researchers to develop clustering approaches particularly for documents [1]-[4]. Since similarity measure is an important component in formulating a clustering approach as it directly decides the effectiveness of a clustering algorithm, some studies focus on developing effective document similarity measures. It has been shown that cosine similarity is better than Euclidean norm for measuring the closeness or distance of two document vectors although Euclidean norm works well for many other types of data. Cosine distance based k-means and fuzzy c-means have been proposed in [1] and [5]. Other than using suitable distance or similarity measure of documents in existing algorithms, co-clustering becomes a more popular way for document clustering [2]-[4]. Unlike classic clustering, which treats document as object and word as feature and only generates document clusters, co-clustering treats document and word as two types of objects and generates both document clusters and word clusters. A fuzzy co-clustering approach is proposed in [6] where document membership and word membership are defined for document clusters and word clusters, respectively. Although these clustering approaches tailored for document data give good performance in document categorization, most of them assume that the dataset can be entirely loaded into memory. Since reading from the disk makes the clustering process too time consuming, existing document clustering approaches are unsuitable for handling large document datasets that exceed available memory limit.

With the development of computer and Internet techniques, especially personal computing techniques, tremendous amount of data have been generated in every second. It becomes more easily for the dataset to exceed the limit of main memory, e.g., ten dimensional data with more than 10^{12} objects are unable to be loaded into memory even for high performance computers [7]. More attentions are given to the scalability issue to develop clustering approaches of large data [8]-[14].Some approaches tend to solve the clustering problem of large dataset by sampling [8], [15], [16]. In these approaches, clustering is performed on a reduced dataset containing a subset of samples from the original dataset. The

clustering result of the extracted samples is then extended to label other objects that are not included in the sampled set. For such kind of approaches, the main challenging is how to produce a reduced dataset to contain sufficient information of the original dataset while is small enough to perform clustering efficiently. Another way for handling large datasets is to treat the data as streaming data, which come one by one or subset by subset [12], [13], [17]. In these approaches, the clusters are successively improved based on previously processed data or historical data and newly coming objects. All objects are clustered by a single pass of the whole data. In order to keep as much as possible the information of passed data within the limited memory, data compression or summarization techniques are used in these approaches to get a compact representation of the historical data. In Single pass fuzzy c-means (SpFCM), the dataset is processed chunk by chunk, where a chunk of data is a small subset of objects. For each chunk, only the centroids weighted by the corresponding total fuzzy membership are kept as historical information to be used together with the coming chunk for next round of clustering. The same research group developed another fuzzy c-means based incremental clustering called Online fuzzy c-means (OFCM) [18]. Different from SpFCM, in OFCM, each chunk of data are clustered individually and the final centroids of the whole dataset are obtained by performing clustering on the collection of centroids from all the chunks. Although these incremental clustering approaches have been shown to be effective for dealing with the large-scale problem in many applications, they are not designed with mechanisms to handle high dimensionality problem and thus are low effective for clustering of large document data.

In this paper, we work on incremental fuzzy clustering for large document data by taking care of scalability as well as the ability for dealing with sparsity and high-dimensionality. We first give some analysis to show that when using fuzzy c-means to cluster a set of sparse and high dimensional objects where the number of objects is much smaller than the dimensionality, it may occur that all the objects are assigned to a single cluster, of which the centroid has the smallest norm. This problem causes performance degradation when directly applying fuzzy c-means based incremental methods to documents. Knowing the reason of why existing SpFCM and OFCM fail to give reasonable results for document dataset, we proposed two new methods which are particularly designed for document data. The first method is a direct modification of the fuzzy c-means based incremental clustering by adding in a step to normalize the centroids after they are updated in each iteration. This modified version is shown to be equivalent to incremental clustering with consine-distance based fuzzy c-means. The second method is to use co-clustering to cluster each chunk of data. Specifically, we propose *Single-Pass* and *Online* incremental clustering with weighted fuzzy co-clustering, where objects have different weights. This is important in *Single-Pass* and *Online* methods as the weights summarize how many previously processed objects that each centroid represents. Experimental results on several benchmark document datasets show that our proposed approaches perform much better than the fuzzy c-means based ones especially with small chunk sizes. This demonstrates that the proposed approaches are more favourable for clustering of large document datasets.

In the next section, we review two fuzzy c-means based incremental clustering approaches SpFCM and OFCM in more details and analyse the problem of applying these two approaches to document data in section III. After that we propose our new incremental fuzzy clustering approaches tailored for document data in section IV. Related work is reviewed and discussed in section V. In section VI, we give experimental results on several real-world document datasets. Finally, we conclude this paper in section VII.

II. FCM-BASED INCREMENTAL FUZZY CLUSTERING

The weighted fuzzy c-means which is used in both SpFCM and OFCM is first briefly reviewed and then both *Single-Pass* and *Online* incremental clustering are discussed.

A. Weighted fuzzy c-means (WFCM)

The weighted fuzzy c-means (WFCM) extends the original fuzzy c-means (FCM) to consider different weights for each of the objects so that objects with large weights play more important roles in clustering than those with small weights. For a dataset $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ to be clustered into k clusters, the objective of WFCM is to minimize the following function

$$J_{WFCM} = \sum_{c=1}^{k} \sum_{i=1}^{n} w_i u_{ci}^m d_E(\mathbf{x}_i, \boldsymbol{\delta}_c) \tag{1}$$

where u_{ci} is the fuzzy membership of object i with respect to cluster c, and

$$d_E(\mathbf{x}_i, \boldsymbol{\delta}_c) = \|\mathbf{x}_i - \boldsymbol{\delta}_c\|^2 \tag{2}$$

is the Euclidean distance between $\mathbf{x}_i$ and $\boldsymbol{\delta}_c$, i.e., the centroid of cluster c. The update equation of membership of WFCM in (1) is the same as that of FCM

$$u_{ci} = \frac{d_E(\mathbf{x}_i, \boldsymbol{\delta}_c)^{-1/m-1}}{\sum_f d_E(\mathbf{x}_i, \boldsymbol{\delta}_f)^{-1/m-1}} \tag{3}$$

and the update equation of centroid is as below

$$\boldsymbol{\delta}_c = \frac{\sum_{i=1}^{n} w_i u_{ci}^m \mathbf{x}_i}{\sum_{i=1}^{n} w_i u_{ci}^m} \tag{4}$$

Next, we discuss how WFCM is used in two ways for incrementally clustering large data.

B. Single-Pass incremental clustering

One way to handle a large dataset is to process the data in small subsets called chunks sequentially and only one pass of the dataset is needed to cluster all the objects. This is called as *Single-Pass* method for incremental clustering. An important characteristic of SpFCM, the fuzzy c-means based Single Pass clustering is to use the centroids as compact information to represent the data have been processed so

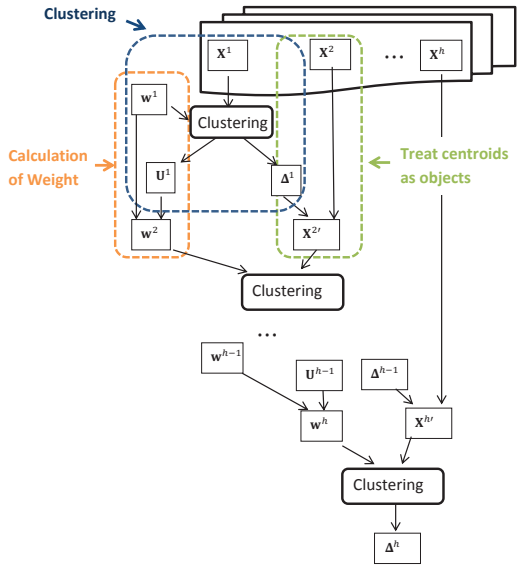

Fig. 1: *Single-Pass* incremental clustering

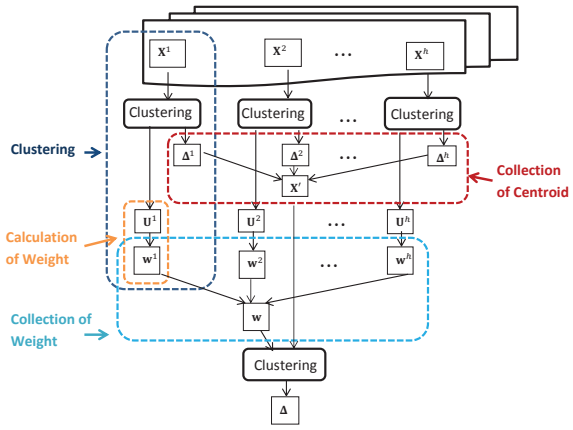

Fig. 2: *Online* incremental clustering

far. The centroids of the previous chunk are combined with the subsequent chunk to form the data for clustering. Each centroid is weighted according to how many objects it represents. As illustrated in Figure 1, *Single-Pass* incremental clustering has three main concepts namely the clustering algorithm used to produce clusters of each chunk of data, the input data to the clustering algorithm, and the weights of objects. Some details are further given below.

- Clustering. WFCM is used as the clustering method in SpFCM. As reviewed early, WFCM takes $(\mathbf{w}, \mathbf{X})$ as input, and produces the membership matrix $\mathbf{U}$ and the k centroids $\mathbf{\Delta} = [\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_k]$. Here $\mathbf{X}$ is a set of s dimensional objects and $\mathbf{w}$ is the vector recording the weights of each object.
- Data. The target dataset is split into small chunks, $\mathbf{Y} = \{\mathbf{X}^1 \cup \mathbf{X}^2 \ldots \cup \mathbf{X}^h\}$. The clustering of the first chunk is only performed on the objects of this chunk. For chunk $t > 1$, the k centroids obtained in the previous chunk is combined with the coming chunk of data for clustering, i.e., $\mathbf{X}^{t'} \leftarrow [\mathbf{\Delta}^{t-1}, \mathbf{X}^t]$.
- Weight. For the first chunk, $\mathbf{w}^1 = \mathbf{w}_{data}$. For chunk $t > 1$, $\mathbf{w}^t = [\mathbf{w}^{t-1}_{center}, \mathbf{w}_{data}]$. The weights for all the n_t objects in every loaded chunk of data are equal to 1, i.e., $\mathbf{w}_{data} = [1, 1, \ldots, 1]^T$. For each centroid, the weight is calculated as a weighted sum of memberships related to the corresponding cluster. Specifically, for $t = 1$, $\mathbf{w}^t_{center} = \mathbf{U}^t \mathbf{w}_{data}$ and for $t > 1$, it consists of the weighted sum of memberships of the centroids of the $t-1$th chunk and the total membership of the objects of the $t-1$th chunk, i.e., $\mathbf{w}^t_{center} = \mathbf{U}^t[\mathbf{w}^{t-1}_{center}, \mathbf{w}_{data}]$.

C. Online incremental clustering

Different from *Single-Pass*, which sequentially processes the data and make modification of the centroids gradually with the new coming data, the *Online* incremental clustering is a *distribute-and-ensemble* method. It individually processes each chunk and then ensembles the centroids of each chunk to give the final centroids. The overall structure of *Online* method is illustrated in Figure 2. Since clustering is performed for each chunk individually, parallel computing can be directly applied to the *Online* method. The centroid ensemble process is also formulated as weighted clustering. Specifically, in the ensemble phase, each centroid is treated as a weighted object, and the $k \times h$ centroids of all chunks form the dataset to be clustered, i.e., $\mathbf{X}' = [\mathbf{\Delta}^1, \ldots, \mathbf{\Delta}^h]$, where each $\mathbf{\Delta}$ is an $s \times k$ matrix representing the k centroids of a chunk. Similar to *Single-Pass*, the weight of each centroid is calculated as the sum of memberships of the corresponding cluster, i.e., $w^h_c = \sum_i u^h_{ci}$, and the weight vector is formed as $\mathbf{w} = [\mathbf{w}^1, \ldots, \mathbf{w}^h]$, with $\mathbf{w}^t = [w^t_1, \ldots, w^t_k]$. It is worthy to notice that the centroids have the same dimensionality as the objects of the original dataset. In other words, if the original dataset is high dimensional, the centroid ensemble task is also a high dimensional data clustering problem.

III. Problem of using SpFCM and OFCM for sparse and high dimensional data

Although SpFCM and OFCM perform well for many large scaled datasets, they may fail to produce reasonable results when applied to document data with a small chunk size. Next we give more details to discuss this problem.

According to (3), the cluster assignment of WFCM is decided by the object-to-centroid distance $d_E(\mathbf{x}_i, \boldsymbol{\delta}_c)$, which is calculated as

$$d_E(\mathbf{x}_i, \boldsymbol{\delta}_c) = \|\mathbf{x}_i - \boldsymbol{\delta}_c\|^2 = \|\mathbf{x}_i\|^2 + \|\boldsymbol{\delta}_c\|^2 - 2\mathbf{x}_i^T \boldsymbol{\delta}_c \quad (5)$$

As the first term of (5) is constant for a given $\mathbf{x}_i$, only the other two terms decide the assignment of an object. For the convenience of discussion, let us consider hard assignment with equal weights, i.e., each object is only assigned to the cluster c with the smallest distance, i.e., $c = \arg\min_f d_E(\mathbf{x}_i, \boldsymbol{\delta}_f)$, and $\mathcal{X}_c$ denotes the set of objects assigned to cluster c. In such a case, $\boldsymbol{\delta}_c$ is a linear combination of all objects that belong to this cluster, i.e.,

$$\boldsymbol{\delta}_c = \frac{1}{|\mathcal{X}_c|} \sum_{j \in \mathcal{X}_c} \mathbf{x}_j \quad (6)$$

With this we have

$$\|\boldsymbol{\delta}_c\|^2 = \boldsymbol{\delta}_c^T(\frac{1}{|\mathcal{X}_c|}\sum_{j\in\mathcal{X}_c}\mathbf{x}_j) = \frac{1}{|\mathcal{X}_c|}\sum_{j\in\mathcal{X}_c}\boldsymbol{\delta}_c^T\mathbf{x}_j \quad (7)$$

When the chunk size or the number of objects of each chunk is small, the number of objects assigned to each cluster is also small, i.e., $|\mathcal{X}_c|$ is small. The extreme case is each cluster only has one object $\mathbf{x}_j$, and thus

$$\|\boldsymbol{\delta}_c\|^2 = \mathbf{x}_j^T\mathbf{x}_j, \quad 2\mathbf{x}_i^T\boldsymbol{\delta}_c = 2\mathbf{x}_j^T\mathbf{x}_i \quad (8)$$

For sparse and high-dimensional document vector $\mathbf{x}_i$, only a very small portion of the elements are non-zero, the number of common words between document $\mathbf{x}_i$ and another document is much smaller than the number of words of each of them. Specifically, we have

$$\mathbf{x}_j^T\mathbf{x}_j \gg \mathbf{x}_j^T\mathbf{x}_i \quad (9)$$

which gives

$$\|\boldsymbol{\delta}_c\|^2 \gg 2\mathbf{x}_i^T\boldsymbol{\delta}_c$$

Therefore, for sparse and high-dimensional data, when the chunk size is small, the distance $d_E(\mathbf{x}_i, \boldsymbol{\delta}_c)$ is dominated by $\|\boldsymbol{\delta}_c\|^2$, which is independent on $\mathbf{x}_i$. This results in all objects being assigned to one cluster, of which the norm of the centroid is the smallest.

If the chunk size is large enough, the number of objects assigned to each cluster $|\mathcal{X}_c|$ is also large enough, then for any j and i,

$$\boldsymbol{\delta}_c^T\mathbf{x}_j \approx \boldsymbol{\delta}_c^T\mathbf{x}_i \quad (10)$$

Based on (10) and (7), we get

$$\|\boldsymbol{\delta}_c\|^2 \approx \boldsymbol{\delta}_c^T\mathbf{x}_i \quad (11)$$

which means the distance $d_E(\mathbf{x}_i, \boldsymbol{\delta}_c)$ is decided by both terms. In other words, the problem of assigning all objects to one cluster is alleviated when the chunk size becomes larger. The above discussion can be extended easily to fuzzy assignments with weighted objects.

IV. Incremental fuzzy clustering of documents

In the above section, we analysed the problem of using WFCM for incrementally clustering of document data. In this section, we propose two methods to handle this problem. One is to modify the existing WFCM in SpFCM and OFCM and the other is to use weighted fuzzy co-clustering in an incremental way. Next, we give the details of each of them.

A. Weighted Hyperspherical fuzzy c-means

To avoid the problem we discussed in the previous section, we normalize all the centroids to unit norm after each iteration

$$\boldsymbol{\delta}_c' = \frac{\boldsymbol{\delta}_c}{\|\boldsymbol{\delta}_c\|} \quad (12)$$

Since all the centroids now have the same norm, the cluster assignment is mainly decided by the third term in (5). For weighted fuzzy c-means

$$\begin{aligned}\|\boldsymbol{\delta}_c\| &= \|\frac{\sum_i u_{ci}^m w_i\mathbf{x}_i}{\sum_i u_{ci}^m w_i}\| \\ &= \sqrt{\sum_j(\frac{\sum_i u_{ci}^m w_i\mathbf{x}_i}{\sum_i u_{ci}^m w_i})^2} = \frac{\sqrt{\sum_j(\sum_i u_{ci}^m w_i\mathbf{x}_i)^2}}{\sum_i u_{ci}^m w_i}\end{aligned} \quad (13)$$

so

$$\boldsymbol{\delta}_c' = \frac{\sum_i u_{ci}^m w_i\mathbf{x}_i}{\sqrt{\sum_j(\sum_i u_{ci}^m w_i\mathbf{x}_i)^2}} \quad (14)$$

Therefore, if we normalize the centroids after each iteration in WFCM, it is actually calculate the normalized centroid $\boldsymbol{\delta}_c'$ by (14). With $\|\boldsymbol{\delta}_c'\| = 1$, the Euclidean distance becomes

$$d_E(\mathbf{x}_i, \boldsymbol{\delta}_c') = \|\mathbf{x}_i\|^2 + 1 - 2\mathbf{x}_i^T\boldsymbol{\delta}_c' \quad (15)$$

Although $\|\mathbf{x}_i\|$ does not affect hard assignment, it may cause the distances to all the centroids very close when $\|\mathbf{x}_i\|^2 \gg \mathbf{x}_i^T\boldsymbol{\delta}_c'$ and makes memberships to all the clusters very similar. A simple way to constrain the value of $\|\mathbf{x}_i\|$ is to normalize each object to unit norm, i.e., $\|\mathbf{x}_i\| = 1$. So the Euclidean distance becomes

$$d_E(\mathbf{x}_i, \boldsymbol{\delta}_c') = 2(1 - \mathbf{x}_i^T\boldsymbol{\delta}_c') = 2d_{cos}(\mathbf{x}_i, \boldsymbol{\delta}_c')$$

where $d_{cos}(\mathbf{x}_i, \boldsymbol{\delta}_c')$ is the cosine distance with $\|\mathbf{x}_i\| = \|\boldsymbol{\delta}_c'\| = 1$. This means that if the document vectors are normalized to unit length, and we normalize the centroids after they are updated in each iteration, the weighted fuzzy c-means becomes a weighted cosine-distance based fuzzy c-means. The cosine distance based fuzzy c-means called Hyperspherical Fuzzy C-Means (HFCM) was first proposed in [5]. So we call its weighted version Weighted Hyperspherical Fuzzy C-Means (WHFCM).

B. Weighted fuzzy co-clustering

Other than using cosine distance in FCM, we also consider to use fuzzy co-clustering to cluster each chunk of data. Unlike classical clustering where only documents are the target objects to be clustered, both documents and words are clustered in co-clustering. Based on the formulation of Fuzzy Co-clustering of Documents and Keywords(FCoDoK) [6], here we propose the Weighted Fuzzy Co-clustering of Documents and Keywords(WFCoDoK). The objective of WFCoDoK is to maximize the value of the following function

$$\sum_{c=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{s} w_i u_{ci} v_{cj} x_{ij} - \frac{\lambda_u}{2}\sum_{c=1}^{k}\sum_{i=1}^{n} u_{ci}^2 - \frac{\lambda_v}{2}\sum_{c=1}^{k}\sum_{j=1}^{s} v_{cj}^2 \quad (16)$$

subject to

$$\sum_{c=1}^{k} u_{ci} = 1 \ \ \forall i = 1,\dots n, \quad \sum_{j=1}^{s} v_{cj} = 1 \ \ \forall c = 1,\dots,k \quad (17)$$

In (16), u_{ci} is the membership of the ith document in cluster c, v_{cj} is the membership of the jth word in cluster c, and w_i is the weight of the ith document. The first term decides the criterion for measuring the quality of clusters. Maximizing this term requires documents and words that co-occurred frequently to be assigned to the same cluster. The other two terms are used for regularization, and λ_u and λ_v are the regularization weights for u and v, respectively.

Similar as in [6], the update rules for u and v are derived as

$$u_{ci} = \frac{1}{\lambda_u} G_{ci} + \frac{1}{k}(1 - \frac{1}{\lambda_u} \sum_{f=1}^{c} G_{fi}) \tag{18}$$

$$v_{cj} = \frac{1}{\lambda_v} H_{cj} + \frac{1}{m}(1 - \frac{1}{\lambda_v} \sum_{h=1}^{s} H_{ch}) \tag{19}$$

with

$$G_{ci} = \sum_{j=1}^{s} w_i x_{ij} v_{cj}; \quad H_{cj} = \sum_{i=1}^{n} w_i x_{ij} u_{ci} \tag{20}$$

Like WFCM, iterative algorithm is used for WFCoDoK to update u_{ci} and v_{cj} successively in an alternating manner. During each iteration, negative values of u and v may appear. A simple way to deal with such kind of situation is to set negative values to 0 and re-normalize u and v to ensure the summation constraints of each are still hold. This strategy is used here as it is shown in [6] to work well in practice. A more complex solution that guarantees non-negative values of u and v can be derived with the Karush-Kuhn-Tucker (KKT) conditions.

As given in (17), the summations of u and v are constrained in different ways. The document membership u is defined in the same way as the membership of fuzzy c-means, which requires that for each document, the sum of its memberships in all the clusters is 1. Differently, the word membership is constrained in a way that for each cluster, the sum of memberships of all the words is 1. We may say that u is the assignment distribution of a document over the k clusters, which reflects the relative degree of belonging of a document in one cluster compared to other clusters; while v is the representativeness distribution over all the words for each cluster, which captures the relative importance of a word compared to other words with respect to the same cluster. In a heuristic fuzzy co-clustering algorithm proposed in [19], v is constrained in the same way as u, and the updating of u and v are derived to maximize two different objective functions rather than one.

The main steps of WFCoDoK-based *Single-Pass* and WFCoDoK-based *Online* incremental clustering are the same as shown in Figure 1 and Figure 2 by using WFCoDoK as the clustering algorithm.

V. Related Work

Document clustering is a challenging task due to the high dimensionality problem. Classic clustering approaches such as k-means and fuzzy c-means are not able to give satisfied performance on document data. In [1], Dhillon and Modha proposed to cluster documents in a hypersphere where the distance between two vectors is the angle between them. The difference between this spherical clustering and the classic k-means algorithm is that different metrics are used to measure the distance between a document and the centroids. K-means uses Euclidean norm while spherical clustering uses inner product. Since in spherical clustering, all document vectors and centroids are normalized to unit length, inner product is equivalent to cosine similarity. A fuzzy c-means approach with cosine distance is proposed in [5]. Other than using suitable distance or similarity measure for document in existing algorithms, clustering has been formulated in many other new ways to handle the high dimensionality problem. Co-clustering is one popular way for document clustering [2], [3], [4]. Unlike classic clustering which only generates document clusters by treating document as object and word as feature, co-clustering simultaneously clusters document and word. It generates both document clusters and word clusters by treating document and word as two types of objects. In the fuzzy co-clustering approach proposed in [6], both document membership and word membership are defined to represent document clusters and word clusters, respectively. Promising results have been reported in these studies which show the ability of co-clustering for handling document data. However, most of these approaches assume that the document data can be entirely loaded into memory and this causes scalability and time efficiency issue when loading in the target document dataset as a whole requires more memory than available.

Clustering of large data has attracted a lot of attentions as the scale of data increased tremendously these years in many data mining applications. Some approaches use various sampling techniques to generate a reduced dataset for clustering and then extend the clustering result to label other objects [8], [15], [16]. The CLARA algorithm proposed in [8] is a random sampling based k-medoids approach. Although random sampling is fast and easy to implement, it is not sure whether the sampled set cover enough information for clustering purpose. To ensure the sampled dataset capture the overall nature of the original dataset, statistics-based progressive sampling is used in [15], [16] for fuzzy clustering of large data. Other than performing clustering on a subset of samples, some other approaches process the dataset sequentially and use compact ways such as representatives to record the previously processed data. The key idea is to keep sufficient statistics of previously processed data and compared to store the original data, it requires much less space to store the statistical information. BIRCH [10] and CURE [11] are two hierarchical clustering where the data is scanned to gradually update the hierarchical tree. A set of heuristic rules and thresholds are used to decide whether the new object should be discarded or added as a new cluster or used for updating any existing clusters.

Single pass partitioning clustering are studied in [12], and [13]. Both of them are based on k-means and only require

single scan of the dataset. In [12], a two-step compression strategy is used to selectively discard some objects and compress the left ones using fine clusters. The approach of [13] is a special case of [12], where all the processed points are discarded and only the centroids weighted by the number of objects assigned to them are kept as a compressed form of the original data. Experimental results in [13] show that this simple single pass approach gives comparable results with the one using complex compression techniques and is more efficient. The approach proposed in [14] is similar to [13]. In [14], the data is assumed to come in chunks and each chunk is clustered with a Local Search algorithm. After clustering, each chunk of data is summarized by the weighted centroids and the data are discarded to free memory. Single pass fuzzy c-means (SpFCM) is proposed in [17], where the weight of each centroid is calculated by the fuzzy memberships rather than the number of objects as in [13], [14]. Other than single pass approaches, an Online fuzzy c-means (OFCM) is proposed in [18]. Different from single pass approaches, in OFCM, each chunk of data is clustered individually and the centroids of all the chunks are gathered for another round of clustering to produce the final cluster centroids. In a recent work [7], the authors studied fast kernel fuzzy c-means by approximation and applied this kernelized version in three types of incremental clustering, namely sampling-based, single pass and online methods. However, kernelized version requires more time to calculate the kernel matrix and its performance is highly dependent on the kernel used. Experimental study of [7] show that compared to the fuzzy c-means based incremental methods, the kernelized version with Gaussian kernel gave worse results.

VI. Experimental results

In this section, we give experimental study of incremental clustering approaches for document categorization. The purpose is to show that the proposed approaches outperform existing SpFCM and OFCM and thus are more favourable for handling large document data.

A. Document datasets

Four benchmark document datasets extracted from different sources are used in our experimental study of incremental clustering. Some statistics of each dataset is summarized in Table I. *Multi5* is from the 20Newsgroups [20]. The total collection of 20Newsgroups contains approximately 20,000 newsgroup articles collected from 20 different newsgroups. *Multi5* consists of around 100 documents from each of the five topics. Dataset *tr12* is derived from the TREC-5, TREC-6 and TREC-7 collections [1]. The *la2* dataset is part of the TREC-5 collection and contains news articles from the Los Angeles Times. The dataset *k1a* was from the WebACE project (WAP) [21], where each document corresponds to a web page listed in the subject hierarchy of Yahoo!. Stop-word removing and stemming are applied as preprocessing. Each term of a document is weighted by *tf-idf* [22]. Each document is normalized to unit length, i.e., $\|\mathbf{x}\|_2 = 1$. Among the five dataset, *Multi5* is relatively easy to cluster as its topics are well separated with balanced size, and only the top 1000 words with the largest mutual information are selected as features. Other three datasets are more difficult for the task of clustering as their dimensionality are very high, the size of clusters are very different and the topics are not well separated.

[1]Text retrieval conference, http://trec.nist.gov

TABLE I: Summary of datasets

Dataset	Source	#documents	#words	#topics	balance
Multi5	20Newsgroups	494	1,000	5	0.95
tr12	TREC	313	5,804	8	0.10
la2	LA Times (TREC)	3,075	12,432	6	0.27
k1a	WebACE	2340	21,839	20	0.018

B. Evaluation

As in [7], we use Adjusted Rand Index (ARI) to evaluate the performance of clustering algorithms. Rand Index is a measure of agreement between two partitions and the adjusted form is adjusted for chance. For a dataset with n objects, assume n_{ij} is the number of objects in common between the ith cluster produced by an algorithm and the jth ground-truth category, ARI is calculated as

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{p_i}{2}\sum_j \binom{q_j}{2}/\binom{n}{2}}{\frac{1}{2}(\sum_i \binom{p_i}{2} + \sum_j \binom{q_j}{2}) - \sum_i \binom{p_i}{2}\sum_j \binom{q_j}{2}/\binom{n}{2}} \tag{21}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient and

$$p_i = \sum_j n_{ij}; \quad q_j = \sum_i n_{ij} \tag{22}$$

C. Experiment setting

We evaluate the performance of the proposed SpHFCM, SpFCoDoK, OHFCM, and OFCoDoK, where the first two are Single pass incremental clustering with WHFCM and WFCoDoK, respectively, and the last two are Online incremental clustering with WHFCM and WFCoDoK, respectively. Their results are compared with existing SpFCM and OFCM. For each dataset, all the algorithms are run with five different sample rates from 5% to 50%. For example, a sample rate of 5% means the dataset is randomly split into 20 chunks with each has 5% documents of the total. For each sample rate, we performed 50 times of random split and all the algorithms are run on each to get 50 clustering results. We report the mean and standard deviation of the 50 trials. The results of each dataset with corresponding non-incremental algorithms, i.e., 100% sample rate, are also reported for comparison. We tried different values of parameters for each algorithm and find Tu=0.001, Tv=0.01 for WFCoDoK and m=1.01 for WHFCM and WFCM give good results.

D. Initialization and assignment

In *Single-Pass*, the clustering of the current chunk is initialized with the centroids of the previous chunk. This way of initialization may also be used in a *Online* method

to increase convergence speed although each chunk may be initialized randomly in a *Online* method. We compare the results of each of the three *Online* incremental clustering approaches with these two types of initialization.

After the final centroids are obtained, cluster assignment of each object can be obtained sequentially with the update equation of u based on these centroids. This step is similar to prediction of cluster label with the given model, i.e., the final centroids. For each clustering approach, we use the corresponding update equation of u and then get the hard clusters by labelling each object to the cluster with the largest membership.

E. Results and discussions

The results of the three *Single-Pass* incremental fuzzy clustering approaches and the three *Online* incremental fuzzy clustering approaches are given in Table II and Table III, respectively. The results of *Online* incremental clustering in Table III are produced based on initialization with the result of the previous chunk.

From these two tables, we have the following observations. First, for both *Single-Pass* and *Online*, it is seen that WFCoDoK performs the best in almost all the cases, i.e., for all the datasets with all the sample rates. The consistent good performance of using fuzzy co-clustering approach to cluster each chunk clearly shows that the formulation of co-clustering is more effective for handling document data than traditional clustering. Second, we found that although the non-incremental FCM gives comparable or slightly worse results with HFCM for three out four datasets, the performance of incremental FCM namely SpFCM and OFCM degrades largely especially when the sample rate is small, such as 5%, 10% and 25%. This confirms that WFCM fails to perform well for sparse document data with a small chunk size as being analysed in Section III. On the other hand, the success of the two WHFCM-based incremental clustering approaches to maintain their performance to a reasonable level with small sample rates demonstrates the effectiveness of the proposed modification. Third, comparing the *Single-Pass* and *Online* incremental clustering with the same fuzzy clustering, it shows that *Online* is generally better than *Single-Pass* for WHFCM and WFCoDoK. The difference for WFCM is less significant.

For *Online* incremental clustering where each chunk is clustered independently, other than using the centroids of the previous chunk for initialization of the current chunk, we may also use random initialization for each chunk. Figure 3 shows the comparison of the three *Online* approaches with these two ways of initialization. It is shown that the results of WFCM at small sample rates especially with the rates that are less than 10% are improved significantly when random initialization is used. This is because the result of FCM of the first chunk might be a bad one due to its problem for handling small set of samples. Using the result of the previous chunk for initialization of the next chunk may pass on the negative impact while using random initialization for each chunk is a simple way to alleviate the negative impact. However, random initialization does not bring benefit for the other two approaches. In fact, random initialization is worse for WHFCM. Since WHFCM is able to produce reasonable results with a small sample rate, the result of the previous chunk is better than a random one, and thus produces improved results. The results of *Online* FCoDoK with two ways of initialization are not different too much, which indicates that this approach is not very sensitive to initialization.

TABLE II: ARI of *Single-Pass* incremental clustering

(a) Multi5

Sample size	SpFCM	SpHFCM	SpFCoDoK
5%	0.15 ± 0.11	0.43 ± 0.09	0.65 ± 0.11
10%	0.30 ± 0.10	0.41 ± 0.09	0.73 ± 0.11
25%	0.52 ± 0.12	0.46 ± 0.13	0.80 ± 0.07
35%	0.53 ± 0.14	0.51 ± 0.13	0.80 ± 0.08
50%	0.59 ± 0.13	0.61 ± 0.13	0.82 ± 0.07
batch	0.74 ± 0.12	0.74 ± 0.12	0.85 ± 0.04

(b) tr12

Sample size	SpFCM	SpHFCM	SpFCoDoK
5%	0.09 ± 0.08	0.37 ± 0.11	0.46 ± 0.10
10%	0.16 ± 0.11	0.35 ± 0.07	0.51 ± 0.09
25%	0.29 ± 0.10	0.36 ± 0.10	0.53 ± 0.05
35%	0.31 ± 0.10	0.35 ± 0.09	0.53 ± 0.05
50%	0.37 ± 0.13	0.39 ± 0.09	0.51 ± 0.05
batch	0.43 ± 0.10	0.45 ± 0.10	0.51 ± 0.02

(c) la2

Sample size	SpFCM	SpHFCM	SpFCoDoK
5%	0.08 ± 0.06	0.35 ± 0.07	0.48 ± 0.07
10%	0.09 ± 0.05	0.36 ± 0.08	0.50 ± 0.08
25%	0.18 ± 0.05	0.43 ± 0.07	0.54 ± 0.03
35%	0.19 ± 0.05	0.46 ± 0.06	0.54 ± 0.02
50%	0.21 ± 0.04	0.48 ± 0.06	0.54 ± 0.02
batch	0.26 ± 0.03	0.50 ± 0.06	0.54 ± 0.02

(d) k1a

Sample size	SpFCM	SpHFCM	SpFCoDoK
5%	0.05 ± 0.04	0.35 ± 0.09	0.37 ± 0.02
10%	0.15 ± 0.07	0.33 ± 0.06	0.44 ± 0.05
25%	0.21 ± 0.08	0.35 ± 0.08	0.39 ± 0.08
35%	0.24 ± 0.05	0.34 ± 0.06	0.38 ± 0.08
50%	0.27 ± 0.05	0.35 ± 0.05	0.40 ± 0.07
batch	0.30 ± 0.05	0.38 ± 0.09	0.41 ± 0.07

VII. Conclusion

In this paper, we study the problem of document categorization using *Single-Pass* and *Online* incremental fuzzy clustering. We first analyse the problem of directly using existing incremental fuzzy clustering approach to handle document data, which is sparse and high dimensional. Based on the analysis, we presented a way to modify the existing fuzzy c-means based incremental clustering by normalizing the centroids to unit length after they are updated in each iteration. This modified algorithm is shown to be equivalent to incremental clustering with cosine distance based fuzzy c-means. We also presented weighted fuzzy co-clustering for handling document datasets incrementally. Experimental results of real-world document datasets show the great potential

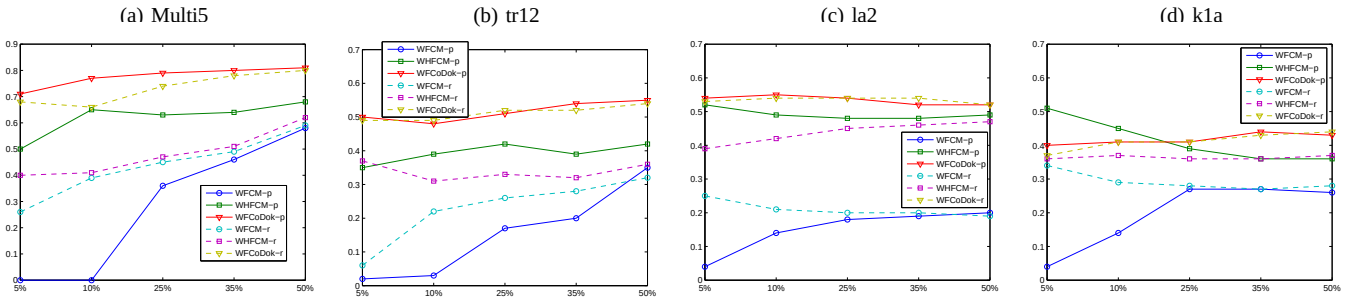

Fig. 3: Comparison of the three *Online* approaches with different initializations. "-p" denotes initialization with centroids of the previous chunk, and "-r" denotes random initialization.

TABLE III: ARI of *Online* incremental clustering

(a) Multi5

Sample size	OFCM	OHFCM	OFCoDoK
5%	0.00 ± 0.00	0.50 ± 0.11	0.71 ± 0.05
10%	0.00 ± 0.00	0.65 ± 0.10	0.77 ± 0.03
25%	0.36 ± 0.13	0.63 ± 0.14	0.79 ± 0.05
35%	0.46 ± 0.17	0.64 ± 0.11	0.80 ± 0.05
50%	0.58 ± 0.13	0.68 ± 0.12	0.81 ± 0.08
batch	0.74 ± 0.12	0.74 ± 0.12	0.85 ± 0.04

(b) tr12

Sample size	OFCM	OHFCM	OFCoDoK
5%	0.02 ± 0.03	0.35 ± 0.09	0.50 ± 0.05
10%	0.03 ± 0.04	0.39 ± 0.09	0.48 ± 0.05
25%	0.17 ± 0.11	0.42 ± 0.09	0.51 ± 0.05
35%	0.20 ± 0.10	0.39 ± 0.09	0.54 ± 0.06
50%	0.35 ± 0.14	0.42 ± 0.10	0.55 ± 0.06
batch	0.43 ± 0.10	0.45 ± 0.10	0.51 ± 0.02

(c) la2

Sample size	OFCM	OHFCM	OFCoDoK
5%	0.04 ± 0.08	0.52 ± 0.07	0.54 ± 0.02
10%	0.14 ± 0.06	0.49 ± 0.06	0.55 ± 0.02
25%	0.18 ± 0.05	0.48 ± 0.06	0.54 ± 0.04
35%	0.19 ± 0.06	0.48 ± 0.06	0.52 ± 0.04
50%	0.20 ± 0.06	0.49 ± 0.05	0.52 ± 0.04
batch	0.26 ± 0.03	0.50 ± 0.06	0.54 ± 0.02

(d) k1a

Sample size	OFCM	OHFCM	OFCoDoK
5%	0.04 ± 0.05	0.51 ± 0.05	0.40 ± 0.08
10%	0.14 ± 0.07	0.45 ± 0.07	0.41 ± 0.07
25%	0.27 ± 0.07	0.39 ± 0.08	0.41 ± 0.08
35%	0.27 ± 0.05	0.36 ± 0.06	0.44 ± 0.06
50%	0.26 ± 0.06	0.36 ± 0.04	0.43 ± 0.07
batch	0.30 ± 0.05	0.38 ± 0.09	0.41 ± 0.07

of the proposed incremental fuzzy clustering approaches for handling large document data.

REFERENCES

[1] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Mach. Learn.*, vol. 42, pp. 143–175, 2001.

[2] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. SIGKDD*, 2001, pp. 269–274.

[3] W. Xu and Y. Gong, "Document clustering by concept factorization," in *Proc. ACM SIGIR*, 2004, pp. 202–209.

[4] A. Banerjee, I. Dhillon, J. Ghosh, S. Meruguand, and D. S. Modha, "A generalized maximum entropy approach to bregman coclustering and matrix approximation," in *Proc. ACM SIGKDD*, 2004, pp. 509–514.

[5] M. E. S. Mendes and L. Sacks, "Evaluating fuzzy clustering for relevance-based information access," in *Proc. IEEE-FUZZ*, 2003, pp. 648–653.

[6] K. Kummamuru, A. Dhawale, and R. Krishnapuram, "Fuzzy co-clustering of documents and keywords," in *Proc. IEEE-FUZZ*, 2003, pp. 772–777.

[7] T. Havens, J. Bezdek, C. Leckie, L. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Trans. Fuzzy Syst.*, vol. 20, pp. 1130–1146, 2012.

[8] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. New York: John Wiley and Sons, 1990.

[9] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. VLDB conference*, 1994, pp. 144–155.

[10] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD*, 1996, pp. 103–114.

[11] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. ACM SIGMOD*, 1998, pp. 73–84.

[12] P. Bradley, U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Proc. SIGKDD*, 1998, pp. 9–15.

[13] F. Farnstrom, J. Lewis, and C. Elkan, "Scalability for clustering algorithms revisited," in *ACM SIGKDD Explorations*, 2000, pp. 51–57.

[14] L. ÓCallaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. ICDM*, 2002, pp. 685–694.

[15] N. R. Pal and J. C. Bezdek, "Complexity reduction for "large image" processing," *IEEE Trans. Syst., Man, and Cybern. B*, vol. 32, no. 5, pp. 598–611, 2002.

[16] R. Hathaway and J. Bezdek, "Extending fuzzy and probabilistic clustering to very large data sets," *Computational Statistics & Data Analysis*, vol. 51, pp. 215–234, 2006.

[17] P. Hore, L. O. Hall, and D. B. Goldgof, "Single pass fuzzy c means," in *Proc. IEEE-FUZZ*, 2007, pp. 1–7.

[18] P. Hore, L. O. Hall, D. B. Goldgof, and W. Cheng, "Online fuzzy c means," in *Fuzzy Information Processing Society, Annual Meeting of the North American*, 2008, pp. 1–5.

[19] W.-C. Tjhi and L. Chen, "A heuristic-based fuzzy co-clustering algorithm for categorization of high-dimensional data," *Fuzzy Sets Syst.*, no. 4, pp. 371–389, 2008.

[20] K. Lang, "Newsweeder: learning to filter netnews," in *Proc. ICML*, 1995, pp. 331–339.

[21] E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. M. Webace, "WebACE: A web agent for document categorization and exploration," in *Proc. Int. Conf. Autonomous Agents*, 1998, pp. 408–415.

[22] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inform. Process. Manag.*, vol. 24, pp. 513–523, 1988.