Akram and Alamgir *Journal of Big Data* (2022) 9:121
https://doi.org/10.1186/s40537-022-00671-7

Journal of Big Data

RESEARCH Open Access

Distributed fuzzy clustering algorithm for mixed-mode data in Apache SPARK

Abdul Wahab Akram† and Zareen Alamgir*†

†Abdul Wahab Akram and Zareen Alamgir contributed equally to this work

*Correspondence: zareen.alamgir@nu.edu.pk

National University of Computer and Emerging Sciences, Lahore, Pakistan

Abstract

Fuzzy clustering is an invaluable data mining technique that allows each data point to belong to more than one cluster with some degree of membership. It is widely employed in exploratory data mining to discover overlapping communities in social networks, find structure in spectral data, and capture user interests in recommendation systems. Nowadays, the variety and volume of data are increasing at a tremendous rate. Data is power; the massive data, along with an effective technique, can unravel valuable information. The existing fuzzy clustering algorithms do not perform well on massive heterogeneous datasets. Processing an enormous amount of data is beyond the capacity of a single processor. The need of the hour is to develop fuzzy clustering techniques that can work on a distributed framework for Big Data processing and can handle heterogeneous data. In this research, we evaluate the performance of the recently proposed algorithm for the Fuzzy clustering of mixed-mode data FCMD-MD (D'Urso and Massari in Inf Sci 505:513–534, 2019) with different real-world datasets. We develop a distributed FCMD-MD, a fuzzy clustering algorithm for mixed-mode data in Apache SPARK. The experimental results show that the algorithm is scalable, performs well in a distributed environment, and clusters enormous heterogeneous data with high accuracy. We also compared the performance of distributed FCMD-MD and the distributed k-medoid algorithm.

Introduction

Clustering is one of the most widely used techniques in exploratory data mining for discovering groups of objects with similar behavior or traits. Currently, it is extensively used in data preprocessing, customer segmentation, data partitioning, outlier detection, and data analyses. It helps to learn useful information and extract interesting patterns from real-world data. Clustering is roughly divided into two major categories: hard and soft clustering. In hard clustering, each object belongs to only one cluster at a time, while in soft clustering, an object can belong to more than one cluster simultaneously. Soft clustering, also known as Fuzzy Clustering, is very useful and has widespread applications. Consider a movie recommendation system where we want to cluster users based on their interests. Here, fuzzy clustering is be a better choice because users may be interested in more than one genre and can become frustrated if only one type of content is

recommended. There are several areas where fuzzy clustering is a more suitable way of clustering data.

Numerous clustering algorithms are designed to cater to different data types and distributions. However, the existing traditional clustering algorithms are incapable of dealing with the everchanging demands and dynamics of Big Data. To extract value from the massive data, the clustering technique needs to effectively deal with the volume and variety of the Big data. The real-world datasets are usually mixed-mode; they consist of different features like continuous, categorical, textual, spatial, time series, and geometric. However, the most commonly used clustering algorithms: K-Means, K-Medoid, Gaussian Mixture Models, and DBSCAN work with only one type of feature (continuous or categorical). K-means works with numeric features, while K-Mode (an extension of k-means) works with categorical data. To make these algorithms work with multiple types, data analysts transform all features into a single data type recognized by the algorithm using some dummy encoding schemes [1], but this increases data dimension and hence, the computation cost. Moreover, it also results in the loss of valuable information.

Gower's [2] pioneered the work on mixed data, and since then, many algorithms have been proposed in the literature to cluster mixed-mode data. KAMILA and K-prototype are the two most commonly used algorithms, but they work only for two types of features (Continuous and Categorical) and form crisp clusters. Furthermore, they require that the user explicitly define weights for each type of feature. So there is a need for an algorithm that can effectively generate fuzzy clusters of mixed-mode data without explicitly defining weights. Recently, Urso et al. [3] proposed FCMd-MD, a Fuzzy C-Medoids clustering model for mixed data. It uses the idea of the PAM(Partitioning around medoid) algorithm for fuzzy clustering of mixed-mode data. The algorithm learns the weights of different features by optimizing the objective function. Hence, there is no need to define weights for different types of variables. The algorithm achieves significantly good results but cannot handle Big data.

Clustering and extracting valuable information from a large volume of datasets is not an easy task. It brings many issues to the front: storing and processing enormous data, extracting patterns, and detecting similarities between data objects. The current computing power of a system is not enough to process such an enormous amount of data; either we must increase computing power, utilize supercomputers or shift to another more suitable technology. Stand-alone servers can offer limited computing resources, and these resources cannot meet the current-era requirements. We need distributed computing technology to work on our commodity hardware and perform parallel processing on gigantic volumes of data. Apache Spark is a big data framework that was introduced to overcome the limitations of the traditionally distributed framework. It is much faster, scalable, programmer-friendly, and provides unification. It also provides a scalable machine learning library known as MLlib to fulfill the needs of large enterprises and help scholars in different research areas. Numerous machine learning algorithms are

included in the MLlib library, but few clustering algorithms, such as K-means and its few variants, are currently provided.

The contribution of this research work is multi-fold. First, we rigorously evaluated the performance of the fuzzy clustering techniques: Fuzzy C-means, Fuzzy C-medoids, and FCMD-MD on different real-world mixed-mode datasets. It is observed that the recently proposed FCMD-MD [3] algorithm outperformed the other techniques, but it is very time-consuming and cannot handle large datasets. In this research, we proposed a distributed FCMD-MD algorithm to perform fuzzing clustering of mixed-mode Big Data. Our algorithm is scalable and can effectively handle massive datasets as it is designed in the Apache Spark framework. We compared the time performance of our algorithm with the sequential FCMD-MD algorithm. We also conducted experiments to compare the performance of our distributed FCMD-MD with the distributed fuzzy k-medoid algorithm in the Apache Spark framework. Furthermore, we also show that the FCMD-MD algorithm is designed for fuzzy clustering, so it does not replace the Kamila algorithm developed for Crisp clusters.

The organization of this paper is as follows: "Related work" section presents the related work, and "Apache Spark" section briefly discusses the details of the Spark framework. "Fuzzy clustering of huge heterogeneous data using Apache Spark" section describes the proposed algorithm in detail. The description of datasets and results of computational experiments are given in "Datasets" and "Computational experiments and analysis" sections, respectively. Finally, the last section is the conclusion.

Related work

The work on mixed data started as early as 1971 when Gower [2] introduced a dissimilarity measure for continuous and categorical variables. Gower distance is computed as an average of different partial dissimilarities across the individual features. After the Gower distance, many algorithms were proposed for the hard clustering of mixed data; however, not much work was done for the fuzzy clustering of mixed data.

Huang [4] proposed a variant of the K-means algorithm called K-prototypes for clustering datasets with continuous and categorical features while maintaining the time efficiency of K-means. The algorithm uses Euclidean and Hamming distances for continuous and categorical variables. Furthermore, it employs decision tree induction algorithms to define the clusters, improving the interpretability of clusters. Bertrand et al. [5] derived an algorithm for clustering medical data with multiple features using Gaussian Mixture Model. Ahmad et al. [6] proposed a K-harmonic type algorithm for clustering mixed data which normalizes and discretizes numerical features in a pre-processing set. Foss et al. [7] proposed a KAMILA algorithm for clustering mixed data. It is considered the state-of-the-art algorithm for clustering data having continuous and categorical features. The algorithm is based on K-means and achieves clustering by equitably balancing the contribution of continuous and categorical features. Skabar [8] proposed an

algorithm that uses graph-based random walk clustering of data with mixed attributes without explicitly defining any distance measure.

The work in the field of fuzzy clustering started in 1984 with the development of the Fuzzy C-means algorithm [9], which is a variant of the K-means algorithm and produces fuzzy clusters. After the Fuzzy C-Means algorithm, developing the Fuzzy C-medoids algorithm was not difficult, as, in the traditional K-means algorithm, the centroids are the mean of the given cluster. In contrast, in the K-medoids algorithm, the centroids are actual data points that have the least distance from all the points in a given cluster. Both algorithms tend to minimize the same objective function. The significant difference lies in the selection of medoids, which makes the K-medoid algorithm less sensitive to outliers. Krishnapuram et al. [10] provided a Fuzzy-based implementation of K-medoids. Wang et al. [11] proposed an algorithm that tries to reduce the limitation of initial cluster selection sensitivity of Fuzzy C-Means by selecting initial clusters using entropy. The algorithm works well on arbitrary-shaped clusters. Bezdek et al. [12] gave a probabilistic implementation of fuzzy c-Means and Ulutagay et al. [13] proposed a density-based fuzzy clustering algorithm from the family of DBSCAN.

Most of the work conducted in the area of fuzzy clustering is for one type of attribute. The different algorithms proposed in the literature are tailored for one particular feature type and are incapable of handling real-world datasets with different features [14]. Nguyen et al. [15] proposed an algorithm for the fuzzy partitioning of categorical data. Wang's incremental fuzzy algorithm [16] handles only time-series data, while Urso et al. [14] algorithm forms fuzzy clusters of spatial and temporal data. Few researchers attempted to handle mixed-mode data clustering. Doring et al. [17] proposed a fuzzy clustering approach based on a probabilistic distance measure that uses Mixture Models to cluster data having both continuous and categorical attributes. Urso and Massari [3] developed an algorithm based on the C-medoids clustering model for finding soft clusters in mixed data. The algorithm learns the weights of different features by optimizing the objective function.

Not much work is done on fuzzy clustering of massive mixed data using the latest distributed platforms like Spark. Jha et al. [18] proposed an Apache Spark-based fuzzy clustering algorithm that utilizes kernel Radial Basis Functions (RBF) to discover clusters in high-dimensional genomics data.

Apache Spark

Apache Spark is an open-source big-data processing framework [19]. We have selected SPARK because it is 10 to 100 times faster than Hadoop. It uses the best features of Hadoop, such as HDFS, for the distribution of data across worker nodes and eliminates its shortcomings; hence it is way faster than Hadoop. In the case of an iterative task, Hadoop reads to and fro the disk for each iteration, while Spark's RDD (Resilient Distributed Dataset) enables it to perform multiple iterative tasks in memory. Apache Spark provides high-level functionalities, unlike Hadoop, where a user must understand the

low-level details of the system. Spark supports multiple languages and different libraries like MLlib and GraphX for machine learning and graph manipulation tasks.

Spark Core and data structures

Spark Core is the main engine of Apache Spark, which is responsible for cluster management, job scheduling, cluster management, input–output operations, and fault recovery.

RDD is a fundamental data structure in Apache Spark. It is viewed as a distributed set of elements that makes parallel computation possible on data. RDD is immutable, which means that data inside RDD cannot be altered. RDDs provide fault tolerance using a lineage graph. If an RDD is lost, it has information about how it was created, so it can be recreated. Two types of operation are performed on RDDs: transformation and action. Transformation is an operation that returns a new RDD by modifying the existing one, while the action is an operation that performs the computation on the existing RDD and returns results. Spark performs lazy computations; an RDD is not transformed unless action is called on them. Spark chooses a lazy approach to decide the best possible way to execute an action.

Dataframe in Apache Spark is a distributed collection of data that is organized into columns. It carries the same objective as python's pandas DataFrames and R Dataframes. DataFrames supports different SQL-like operations: aggregate, group, filter and join. Existing RDD or external tables can be easily converted to Spark DataFrame.

Fuzzy clustering of huge heterogeneous data using Apache Spark

This section presents a distributed fuzzy clustering algorithm to efficiently cluster large heterogeneous datasets using the distributed framework Apache Spark. Our algorithm is inspired by the FCMD-MD model recently proposed by Urso et al. [3]. We briefly discuss the FCMD-MD model before presenting our Spark-based distributed FCMD-MD algorithm.

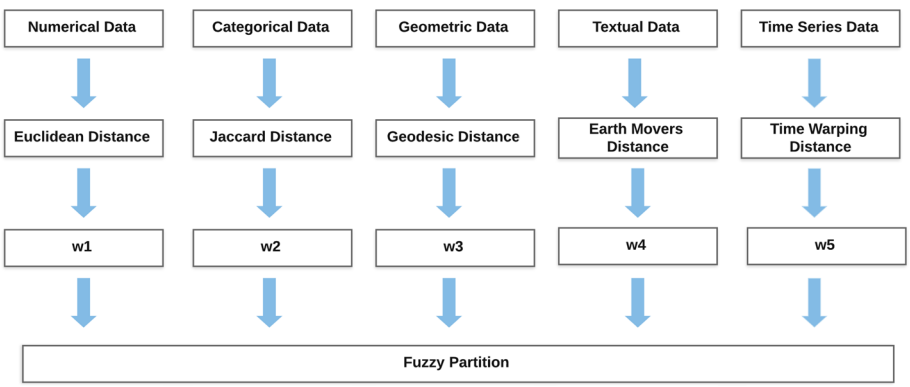

Fig. 1 FCMD-MD algorithm approach [3]

Fuzzy clustering model for mixed-mode data (FCMD-MD)

This technique is based on K-medoid's approach and creates a fuzzy partition of data with mixed features. The algorithm works by optimizing the cost function, the weighted sum of dissimilarities of each feature type. Figure 1 gives an example of the working of the FCMD-MD algorithm for data consisting of five different feature types. The figure presents the underlying idea of the FCMD-MD model; different distance measures and weights are applied to each feature type to compute similarities and create fuzzy partitions. Algorithm 1 shows the complete pseudocode of the FCMD-MD algorithm as proposed in the paper [3].

Let $X_i = \{X_1, \ldots, X_S\}$ be a data point which is a combination of S types of features. Each X_i is a set consisting of one or more variables of a particular type. For example, let us assume that we have two types of variables, hence $S = 2$ and $X_i = \{X_1, X_2\}$. Further, assume that X_1 is a set of two quantitative variables $X_1 = \{X_{11}, X_{12}\}$ and X_2 is a set of three categorical variables, $X_2 = \{X_{21}, X_{22}, X_{23}\}$. Depending on the nature of variables, X_s can be a vector or can have a more complex structure. For example, if it is the combination of continuous variables, then it is a vector and if X_s represents a time series data, then it can be a matrix.

The distance between two data points i and j based on sth feature can be calculated as:

$$ {}_s d_{ij} = d\left(X_{is}, X_{js}\right) $$

${}_s d_{ij}$ can be Euclidean distance in case of continuous variables and Simple matching coefficient measure in case of categorical variables.

Objective function

The FCMD-MD algorithm tends to minimize the objective function. The objective function as defined in [3] is given below:

$$ \sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^{m} d_{ic}^{2} = \sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^{m} \sum_{s=1}^{S} \left(w_s \cdot_s d_{ic}\right)^2 \tag{1} $$

here:

$$ \sum_{c=1}^{C} u_{ic} = 1 \tag{2} $$

$$ \sum_{s=1}^{S} w_s = 1 \tag{3} $$

u_{ic} and w_s are computed by calculating the Lagrangian of the objective functions and are given below (for details, refer to [3]).

$$u_{ic} = \frac{1}{\sum_{c'=1}^{C} \left[\frac{\sum_{s=1}^{S} (w_s \cdot {}_s d_{ic})^2}{\sum_{s=1}^{S} (w_s \cdot {}_s d_{ic'})^2} \right]} \tag{4}$$

$$w_s = \frac{1}{\sum_{s'=1}^{S} \left[\frac{\sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^m \cdot {}_s d_{ic}^2}{\sum_{i=1}^{n} \sum_{c=1}^{C} u_{ic}^m \cdot {}_{s'} d_{ic}^2} \right]} \tag{5}$$

where:

- u_{ic} indicates the membership degree of the ith object in cth cluster,
- w_s indicates the weights associated with sth feature type,
- m represents the fuzziness parameter,
- ${}_s d_{ic}$ represents the distance between ith observation and cth cluster according to sth feature,
- d_{ic}^2 is the overall weighted squared distance between ith observation and cth cluster medoid.

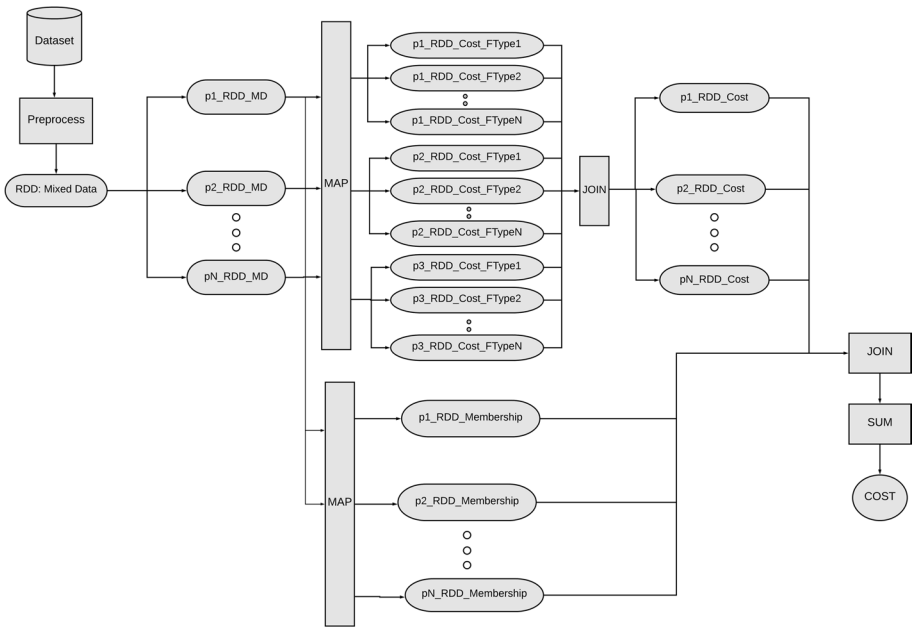

Fig. 2 Distributed FCMD-MD

Algorithm 1 FCMD-MD Algorithm

Fix C and max_iter
$iter \leftarrow 0$
Pick Initial Medoids
repeat
    Store the current Medoids
    Compute U using 4
    Compute W using 5
    for c = 1 to C **do**
        $q \leftarrow argmin_{1 \leqslant i' \leq n} \sum_{i''=1}^{n} u_{i''}^{m} \sum_{s=1}^{S} (w_{s \cdot s} d_{i',i''})$ **return** $\bar{X}_{cs} = X_{qs}$
    $iter \leftarrow iter + 1$
until $\bar{X}_{OLD,S} = \bar{X}_{s,S}$ or iter = max_iter

Distributed fuzzy clustering for mixed-mode data in Apache SPARK

In this section, we present the Spark-based distributed algorithm for fuzzy clustering of mixed-mode data. The input data is pre-processed, partitioned, and persisted in memory as an RDD, Spark's fundamental distributed data structure. Most of the computation is carried out on the persisted RDD to speed up the processing.

Algorithm 4 shows the complete pseudocode of distributed FCMD-MD implemented in Apache Spark. The dataset is pre-processed using Algorithm 2. The algorithm 2 creates a key-value RDD and performs different pre-processing tasks such as handle missing values, remove noise/outliers and normalize the continuous attributes. Furthermore, it arranges the different features according to their type and creates a dictionary with feature type as key and a list of features as attributes.

The initial set of medoids is selected randomly from distributed dataRDD using the function "takeSample". The variables k, W, m, and *medoids* are broadcasted across the cluster as all the executors require them for performing computations. The total cost for the current medoids is computed using Algorithm 3. Figure 2 shows the one complete spark job performed by the distributed version of FCMD. Given the current medoids, the pre-processed data RDD is used to compute distance RDDs for each type of variable that is later joined and transformed to have collective distance RDD. The collective distance RDD is joined with the membership matrix RDD and summed up to get the final cost for current medoids. Depending on the final cost, the swapping of medoids is decided. Several such jobs run in one iteration of the algorithm depending on input data size and the number of clusters, weights for each type of variable, and membership matrix are recomputed after each iteration.

Algorithm 2 Data Preprocessing

procedure PREPROCESSDATA(*data*)
    $dataRDD \leftarrow data.zipWithIndex()$ ▷ Create key value RDD, key is the index of data point
    $dataRDD \leftarrow dataRDD.map(key, missingValues())$ ▷ Remove noise and missing values
    $dataRDD \leftarrow dataRDD.map(key, normalizeFeatures())$ ▷ Normalize continuous features
    $dataRDD \leftarrow dataRDD.map(key, arrangeFeatures())$ ▷ Function arrangeFeatures() return the dictionary with feature type as key and list of features as values e.g (1385, {'continuous':[0.4,0.2,1.5], 'categorical':['male','single']})
    return *dataRDD*

Algorithm 3 Distributed Distance Calculation

procedure CALCULATEDISTANCE($dataRDD$, $datatypes$, U)
  for $type \in datatypes$ **do**
    $distanceRDD[type] \leftarrow dataRDD.map(key, computeDistance())$ ▷ Compute distance of each data point from medoids using appropriated distance measure according to the feature type.
  $totalDistanceRDD \leftarrow join(distanceRDD)$ ▷ Joins all DistanceRDD
  $cost \leftarrow totalDistanceRDD.join(U).sum()$ ▷ Joins totalDistanceRDD with Membership RDD U, multiplies the values in both RDDs and sums up to have total cost
  return $cost$, $distanceRDD$, $totalDistanceRDD$

Algorithm 4 Distributed FCMD-MD

procedure MAIN($data.csv$, $config.json$, $partitions$ $maxIterations$)
  $config \leftarrow readConfigurationFile(config.json)$
  $data \leftarrow readDataFile(data.csv)$
  m, s, k, datatypes $\leftarrow config.get()$
  $dataRDD \leftarrow preprocessData(data, config)$
  $dataRDD \leftarrow dataRDD.partitionBy(partitions).persist()$
  $medoids \leftarrow dataRDD.takeSamples(k)$
  $W \leftarrow \{key : 1.0/s\ for\ key\ in\ datatypes\}$
  $U \leftarrow dataRDD.map(lambda\ x : (x[0], [1.0/k] * k))$
  Broadcast variables k, W ,m, medoids and datatypes
  $count \leftarrow 0$
  while medoidSwap and count ¡ maxIterations **do**
    $medoidSwap \leftarrow False$
    $cost$, $distanceRDD$, $totalDistanceRDD \leftarrow calculateDistance(dataRDD, datatypes, U)$
    $U \leftarrow totalDistanceRDD.map(lambda\ x : (x[0], update\_membership(x[1])))$
    ▷ update_membership updates the values as per 4
    Update weights W as per 5
    for $i \in medoids$ **do**
      for $j \in dataRDD$ where data point has highest membership value in cluster i **do**
        if $j \notin medoids$ **then**
          $new\_medoids \leftarrow deepcopy(medoids)$
          $new\_medoids[i] \leftarrow j$
        else
          continue
        $new\_cost \leftarrow calculateDistance(dataRDD, datatypes, U)$
        if new_cost $<$ costs **then**
          $cost \leftarrow new\_cost$
          $medoid[i] \leftarrow j$
          $swap \leftarrow True$
    $count \leftarrow count + 1$
  $U.saveAsText()$

Table 1 Overview of datasets

Dataset	Number of observations	Continuous features	Categorical features	Geometric features	Outcome feature
Australian Credit	690	6	8	N/A	Acc (44%), Rej (56%)
Cylinder Bands	516	7	13	N/A	Band (41%), No_Band (59%)
Online shopping intention	12330	10	7	N/A	FALSE (84.5%), TRUE (15.5%)
AirBnB	48895	5	2	2	N/A

Datasets

The datasets are obtained from the UCI Machine Learning Repository, and Kaggle [20, 21]. Table 1 shows the split of attribute types and classes in different datasets. We have considered the dataset of various sizes as we wish to examine the performance of our distributed fuzzy clustering technique and compare it with the sequential FCDM-DM algorithm, which cannot handle large datasets.

Australian Credit

Australian Credit dataset [22] concerns credit card applications; the original names and values of attributes are replaced with meaningless values because of confidentiality purposes. The dataset is interesting as it combines continuous and categorical attributes; there is an almost equal proportion of continuous and categorical attributes.

Cylinder Bands

Cylinder Bands dataset [23] contains information regarding the delay in the printing process. The variables represent different parts of the printing process, such as paper size, ink color, ink type, and cylinder size. The class label describes whether the banding occurred during the process of printing or not. This dataset contains far fewer continuous features than categorical features. This dataset contains many missing values; hence it is prepossessed. The variable that contains more than eight missing values is dropped. If the variable contains 1 to 8 missing values, the corresponding rows are removed from the dataset.

Online shoppers intention (OSI)

OSI dataset [24] represents the user's intention given the combination of mixed types of features; label is a Boolean feature that represents whether the user checks out something or not. This dataset compares the performance of distributed FCMD-MD with its serialized version, as the previous two are not large enough to show the effectiveness of Distributed FCMD-MD. The table shows the split of attribute types and classes.

AirBnB

Airbnb dataset [25] is publicly made available by Airbnb. It shows the rental listing of AirBnB in New York City for the year 2019. The dataset is a combination of numerical, categorical, and geometrical features that are preprocessed to drop some meaningless features.

Computational experiments and analysis

The rigorous computational experiments are conducted to identify the best technique for fuzzy clustering of a mixed-mode dataset in a sequential and distributed mode.

The performance of FCMD-MD on different datasets is reported in this section. First, we conducted experiments to evaluate the FCMD-MD algorithm's effectiveness on Cylinder Bands and Australian Credit datasets and compare it with state-of-the-art fuzzy clustering algorithms. We also compare FCMD-MD performance with a hard-clustering

Table 2 Sequential machine specifications

Specification	Value
Cores	8
Memory	16GB
Operating system	Linux

Table 3 Cluster specifications

Specification	Value
Worker nodes	5
Cores per worker node	8
Memory per worker node	30GB
Operating system	Linux

algorithm, KAMILA designed for mixed-mode data. Furthermore, we compare the sequential FCMD-MD algorithm with the proposed distributed version implemented in Apache SPARK; the time taken by both versions of the algorithm is reported on OSI and AirBnB datasets.

Experimental setup

This section gives details of the experimental setup used to run distributed and sequential versions of FCMD-MD. Table 2 shows the specifications of a machine used to compute the results for the sequential version of FCMD-MD. Table 3 shows the specification of cluster acquired on Google DataProc; out of five worker nodes, one is used as cluster master while the remaining four work as regular worker nodes.

Evaluation metrics

The paper uses two types of evaluation metrics: Purity and Fuzzy Rand Index (FRI). Purity is an external evaluation measure for clustering; it calculates how pure the clusters are by computing the most common class type in each cluster. Purity can be defined for each cluster. It counts the number of points of a majority class, then take the sum over all clusters and divide it by the total number of data points.

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max\left|c_i \cap t_j\right|$$

The fuzzy Rand Index is used as an evaluation metric to analyze the quality of the generated fuzzy clusters. Rand Index is an external evaluation measure that requires true labels. It checks whether each pair of point belong to the same cluster or not; one can see Rand Index as Accuracy:

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

- TP: Two points should belong to the same cluster and our algorithm assigns them to the same cluster
- TN: Two points should belong to different clusters and our algorithm assigns them to different clusters
- FP: Two points should not belong to the same cluster but our algorithm assigns them to the same cluster
- FN: Two points should belong to different clusters, but our algorithm assigns them to different clusters

Fuzzy Rand Index is a fuzzy variant of the Rand Index [26]. Its value ranges from 0 to 1.

Comparison of FCMD-MD with Fuzzy clustering algorithms on mixed-mode data

This section compares the FCMD-MD algorithm with two state-of-the-art fuzzy-based clustering algorithms: Fuzzy C-Means and Fuzzy C-Medoid. The fuzzy rand index is

Table 4 Average Fuzzy Rand Index achieved by FCMD-MD, Fuzzy C-Means and Fuzzy C-Medoids

Dataset	FCMD-MD (FRI) (%)	Fuzzy C-Means (FRI) (%)	Fuzzy C-Medoids (FRI) (%)
Australian Credit	66.40	61.81	61.69
Cylinder Bands	51.60	50	50.02

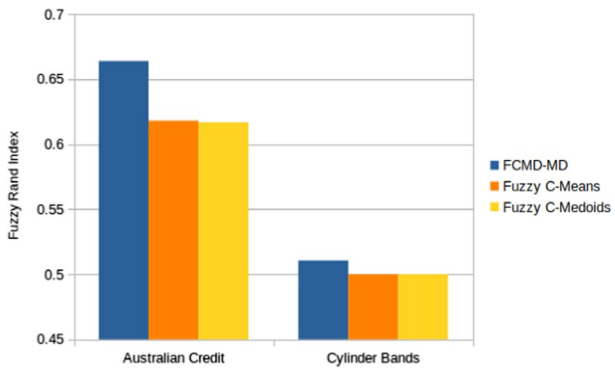

Fig. 3 Comparison of FCMD-MD with Fuzzy C-Means and Fuzzy C-Medoids

Table 5 Time consumption comparison of FCMD-MD with Fuzzy C-Means and Fuzzy C-Medoids

Dataset	FCMD-MD time (s)	Fuzzy C-Means time (s)	Fuzzy C-Medoids time (s)
Australian Credit	29.80	0.018	5.24
Cylinder Bands	39.36	0.03	8.844

used as a performance metric. Both Fuzzy C-Means and Fuzzy C-Medoids algorithms work only with continuous data. Therefore, the categorical features are transformed into continuous features using dummy encoding, increasing the dimensionality of both datasets. Table 4 and Fig. 3 show the results of the comparison of fuzzy clustering techniques. The FCMD-MD algorithm outperformed both Fuzzy C-Means and Fuzzy C-Medoids for both datasets. It achieved 66.4% FRI on the Australian Credit data set, while Fuzzy C-Means and Fuzzy C-Medoids scored relatively less, around 61%.

Table 5 shows the time taken by different techniques. The time consumption of the FCMD-MD algorithm is relatively high compared to other techniques, and this highlights the need for a distributed FCMD-MD algorithm that can speed up the algorithm while maintaining its performance.

Comparison of FCMD-MD with hard-based clustering algorithm (KAMILA)

This section compares the FCMD-MD algorithm with a state-of-the-art hard-based clustering algorithm KAMILA, for continuous and categorical variables. The FCMD-MD technique produces fuzzy partitions of data, so to compare it with KAMILA, the fuzzy partitions are converted to hard partitions by assigning each data point to a cluster with a maximum value in the membership matrix.

Table 6 shows the comparison of average purity achieved by FCMD-MD and KAMILA. It is observed from the results that the FCMD-MD could not outperform KAMILA for hard-based clustering of data. Table 7 shows the time consumed by

Table 6 Average purity achieved by FCMD-MD and KAMILA

Dataset	FCMD-MD (purity) (%)	KAMILA (purity) (%)
Australian Credit	65.27	68.58
Cylinder Bands	62.70	67.80

Table 7 Time consumption comparison of KAMILA and FCMD-MD

Dataset	FCMD-MD time (s)	KAMILA time (s)
Australian Credit	29.80	0.216
Cylinder Bands	39.36	0.234

Table 8 Performance of FCMD-MD on Australian Credit dataset

No.	Distance measure (continuous)	Normalization technique (continuous)	Minimum purity (%)	Average purity (%)	Maximum purity (%)
1	Euclidean	None	55.07	57	70.43
2	Euclidean	Min–Max	58.26	65.27	73.04
3	Euclidean	Standard scalar	55.07	62.40	76.81
4	Manhattan	None	55.05	55.79	58.40
5	Manhattan	Min–Max	55.50	65	74.34
6	Manhattan	Standard scalar	55.50	61.80	76.67

Table 9 Results of different experiment of FCMD-MD on Cylinder Bands dataset

No.	Distance measure (continuous)	Normalization technique (continuous)	Minimum purity (%)	Average purity (%)	Maximum purity (%)
1	Euclidean	None	58.85	62.11	65.52
2	Euclidean	Min–Max	58.85	59.10	61.71
3	Euclidean	Standard scalar	58.85	59.60	61.50
4	Manhattan	None	58.85	61.77	62
5	Manhattan	Min–Max	58.85	62.70	63.80
6	Manhattan	Standard scalar	58.85	59.50	61.50

Table 10 Importance assigned by FCMD-MD to each feature type

Dataset	Continuous features (importance) (%)	Categorical features (importance) (%)
Australian Credit	72.26	27.74
Cylinder Bands	31.78	68.22

Table 11 Job parameters

Parameter	Value
Number of executors	8
Number of cores per executor	4
spark.locality.wait	0
spark.task.cpus	4

Table 12 Time consumed by distributed and sequential version of FCMD-MD

Dataset	Sequential FCMD-MD time (min)	Distributed FCMD-MD time (min)
OSI	447	237
AirBnB	3862	1615

FCMD-MD and KAMILA on both datasets. The time taken by FCMD-MD is greater than KAMILA. It is evident from this experiment that FCMD is not ideal for crisp clusters, and it is not a replacement for a hard-clustering algorithm. It is designed for fuzzy clustering and should only be used when fuzzy clusters are required for mixed-mode data.

Effect of normalization techniques on FCMD-MD clustering

Tables 8 and 9 show the performance of the FCMD-MD algorithm under six different experimental settings. The algorithm is executed ten times in each run the minimum, maximum, and average values of purity are reported. The experimental results show that the algorithm performs best when continuous features are normalized. In experiments 2 and 5, the FCDM-DM performs best on the Australian Credit and Cylinder

Table 13 Iteration vs error (OSI dataset)

Iteration	Error value
1	2997
2	614.92
3	173.46

Table 14 Iteration vs error (AirBnB dataset)

Iteration	Error value
1	664.82
2	0.023

Table 15 Importance/weightage assigned by distributed FCMD-MD to each feature type

Dataset	Continuous features (importance) (%)	Categorical features (importance) (%)	Geometric features (importance)
OSI	73.21	26.79	N/A
AirBnB	42.10	57.89	0.01%

Table 16 Convergence time of distributed FCMD-MD and Fuzzy C-Medoid on OSI dataset

Dataset	Distributed FCMD-MD time (min)	Distributed Fuzzy C-Medoid time (min)
OSI	237	102

Table 17 Fuzzy Rand Index (FRI) scored by distributed FCMD-MD and Fuzzy C-Medoid on OSI dataset

Dataset	Distributed FCMD-MD (FRI)	Distributed Fuzzy C-Medoid (FRI)
OSI	66.67%	61.84%

bands dataset. Note that the continuous features are normalized in both of these best-performing experiments. Table 10 shows the weights assigned by FCMD-MD to each feature type. For the Australian Credit dataset, continuous features got more importance, while categorical features were assigned more weights for the Cylinder Bands dataset.

Distributed FCMD-MD results

We conducted experiments to compare the running time of distributed and sequential FCMD-MD algorithm. The experiments were executed on OSI and AirBnB datasets as they are relatively big. The parameters used to run distributed FCMD-MD on the acquired cluster are shown in Table 11. Table 12 shows the time consumed by both

versions of FCMD-MD. The distributed version outperformed the sequential version with regard to time and is a lot faster which can be further improved depending on the specifications of the cluster and optimal parameters to run the algorithm on a cluster.

Tables 13 and 14 show the convergence of the distributed FCMD-MD on OSI and AirBnB datasets. It is observed that the algorithm's convergence is quite fast, and it converged in 3 iterations on the OSI dataset, while it took only two iterations to converge on the Airbnb dataset. Table 15 shows the weights assigned by the distributed FCMD-MD to each variable on OSI and AirBnB datasets. For the OSI dataset, the continuous features got more importance as compared to categorical features, while the algorithm assigned less than one percent weightage to geometric features for the Airbnb dataset and categorical features got the most importance.

Comparison of the distributed FCMD-MD and Fuzzy C-Medoid

This section compares the performance of the distributed FCMD-MD with the distributed Fuzzy C-Medoid algorithm on the OSI dataset. Table 16 shows the convergence time of both algorithms on the OSI dataset, while Table 17 shows the Fuzzy Rand Index attained by both algorithms. The distributed FCMD-MD algorithm outperformed the Fuzzy C-Medoid; however, it took a bit more time than the distributed fuzzy c-medoid.

Analysis

The FCMD-MD algorithm works significantly well on data with mixed types of features. It outperformed the most commonly used algorithm for fuzzy-based clustering and proved more effective than Fuzzy C-Medoid. The performance of the FCMD-MD algorithm improves when continuous attributes are normalized, as shown in Tables 8 and 9. The basic idea of the FCMD-MD algorithm is based on the K-Medoid algorithm. Hence, it is a bit slow as it inherits all the limitations of K-Medoid. Moreover, it has to perform more work while computing the membership matrix of data points and assigning weights to each feature type. Due to this reason, there was a need to improve its time consumption. Hence, the distributed version of FCMD-MD is proposed in this paper, which shows promising results on different datasets. Even on small-sized datasets, the distributed version reduced the time consumption of the sequential version by half, as shown in Table 12. Another exciting aspect of this algorithm is its fast convergence. As we can see from Tables 13 and 14, the algorithm provides significant improvement with each iteration.

Conclusion and future work

A scalable SPARK-based distributed version of the FCMD-MD algorithm is presented in this research work. The algorithm gave promising results in the fuzzy clustering of data with mixed types of features, and it outperformed the most commonly used fuzzy-based clustering algorithms like Fuzzy C-means and Fuzzy C-medoid. The distributed version of FCMD-MD improves the computation time and can cluster enormous datasets effectively. Future work includes the performance and computation time of distributed FCMD-MD on massive mixed-mode datasets using a large-capacity cluster.

Acknowledgements
Not applicable.

Author contributions
The authors have equal contributions. ZA gave the vision, direction and worked on algorithm development. AWA implemented the code and conducted experiments. The manuscript was co-written by both authors. Both authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The data sets used are publicly available and link included in reference.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 7 April 2022 Accepted: 29 November 2022
Published: 21 December 2022

References

1. Ahmad A, Hasmi S. Generalized Minkowski metrics for mixed feature-type data analysis. IEEE Trans Syst Man Cybern. 1994;24(4):698–708.
2. Gower JC. A general coefficient of similarity and some of its properties. Biometrics. 1971;27:857–71.
3. D'Urso P, Massari R. Fuzzy clustering of mixed data. Inf Sci. 2019;505:513–34.
4. Huang Z. Clustering large data sets with mixed numeric and categorical values. In: Proceedings of the 1st Pacific-Asia conference on knowledge discovery and data mining, (PAKDD); 1997. p. 21–34.
5. Saâdaoui F, Bertrand PR, Boudet G, Rouffiac K, Chamoux A. A dimensionally reduced clustering methodology for heterogeneous occupational medicine data mining. IEEE Trans NanoBiosci. 2015;14(7):707–15.
6. Ahmad A, Hasmi S. K-harmonic means type clustering algorithm for mixed datasets. Appl Soft Comput. 2016;48:39–49.
7. Foss A, Markatou M, Ray A.H. Bonnie. A semiparametric method for clustering mixed data. Mach Learn. 2016;105:419–58.
8. Skabar A. Clustering mixed-attribute data using random walk. Procedia Comput Sci. 2017;108:988–97.
9. Bezdek J, Ehrlich R, Full W. FCM: the fuzzy c-means clustering algorithm. Comput Geosci. 1984;10:191–203.
10. Bezdek J, Ehrlich R, Full W. Low-complexity fuzzy relational clustering algorithms for web mining. IEEE Trans Fuzzy Syst. 2001;9(4):595–607.
11. Su X, Wang X, Wang Z, Xiao Y. An new fuzzy clustering algorithm based on entropy weighting. J Comput Inf Syst. 2010;6(10):3319–26.
12. Pal NR, Pal K, Keller JM, Bezdek JC. A possibilistic fuzzy c-means clustering algorithm. IEEE Trans Fuzzy Syst. 2005;13(4):517–30.
13. Ulutagay G, Nasibov E. Fn-dbscan: a novel density-based clustering method with fuzzy neighborhood relations. In: 8th international conference on application of fuzzy systems and soft computing (ICAFS-2008); 2008. p. 101–10.
14. D'Urso P, De Giovanni L, Disegna M, Massari R. Fuzzy clustering with spatial–temporal information. Spat Stat. 2019;30:71–102. https://doi.org/10.1016/j.spasta.2019.03.002.
15. Mau TN, Huynh V-N. Kernel-based k-representatives algorithm for fuzzy clustering of categorical data. In: 2021 IEEE international conference on fuzzy systems (FUZZ-IEEE); 2021.
16. Wang L, Xu P, Ma Q. Incremental fuzzy clustering of time series. Fuzzy Sets Syst. 2021;421:62–76.
17. Doring C, Borgelt C, Kruse R. Fuzzy clustering of quantitative and qualitative data. In: IEEE annual meeting of the fuzzy information, Vol. 1. IEEE; 2004. p. 84–9.
18. Jha P, Tiwari A, Bharill N, Ratnaparkhe M, Mounika M, Nagendra N. Apache spark based kernelized fuzzy clustering framework for single nucleotide polymorphism sequence analysis. Comput Biol Chem. 2021;92:107454.
19. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I. Apache spark: a unified engine for big data processing. Commun ACM. 2016;59:56–65. https://doi.org/10.1145/2934664.
20. Dua D, Graff C. UCI machine learning repository; 2017. http://archive.ics.uci.edu/ml.
21. Kaggle. https://www.kaggle.com.
22. Australian credit dataset. http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval).
23. Evans B. Cylinder bands dataset; 1995. https://archive.ics.uci.edu/ml/datasets/Cylinder+Bands.
24. Saka CO, Kastro Y. Online shoppers purchasing intention dataset; 2018. http://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset.

25. Dhakar R. Airbnb dataset; 2018. https://www.kaggle.com/ronikdhakar/airbnb-dataset#Airbnb-Dataset.
26. Hullermeier E, Rifqi M, Henzgen S, Senge R. Comparing fuzzy partitions: a generalization of the rand index and related measures. IEEE Trans Fuzzy Syst. 2012;20:546–56. https://doi.org/10.1109/TFUZZ.2011.2179303.

Publisher's Note