

# A novel Split-merge-evolve $k$ clustering algorithm

MingXue Wang  
Network Management Lab, Ericsson  
Ireland  
mingxue.wang@ericsson.com

Vincent Huang  
Ericsson  
Sweden  
vincent.a.huang@ericsson.com

Anne-Marie Cristina Bosneag  
Network Management Lab, Ericsson  
Ireland  
anne.marie.cristina.bosneag@ericsson.com

**Abstract**—Clustering algorithms are used in a large number of big data analytic applications spread across various application domains, including network management. We propose a novel Split-merge-evolve algorithm for clustering data into  $k$  number of clusters. The algorithm randomly divides data into  $k$  clusters initially, then repeatedly splits bad clusters and merges closest clusters to evolve the final clustering result. A key metric during the clustering process of the Split-merge-evolve algorithm is a user chosen or defined clustering quality metric or internal evaluation. The algorithm evolves the clustering result towards the user expected high quality result, although there is no ground truth or labelled data involved during the clustering process. The algorithm design is flexible in its implementation, with various common techniques, such as centroid and connectivity based measures that can be used in its implementation. The algorithm is easy to implement and effective. With 4 datasets, including 2 real life datasets in our experiments, the Split-merge-evolve algorithm performs better than both most commonly used K-means and Agglomerative hierarchical algorithms.

## 1. Introduction

In the last years, machine learning algorithms have continuously become more important in the big data analytic application development space for many organisations, their adoption enabling the introduction of new features such as prediction, anomaly detection, etc [1]. In particular, unsupervised machine learning algorithms are frequently used machine learning algorithms in application feature development, since large amounts of labelled data are unavailable for many use cases. Among the unsupervised learning algorithms, clustering is a very popular type of problem, which groups similar data vectors or objects in the same group or cluster. Hence, data vectors within a cluster have strong similarity between themselves and dissimilarity with other clusters' data vectors.

Clustering has a large number of applications spread across various domains, including network management. For example, in the area of troubleshooting, one idea is to group together millions of network commands, alarms, and various network events having strong relationships between them, based on clustering algorithms [2]. As a result, a group of

various network events can be seen as representing a single network 'incident' - these events are closely related together and also closely related to a same network problem. Network operators are provided with incident information that simplifies the time taken to analyse and resolve problems in their network. They do not need to handle individual alarms or events separately.

There are a large number of clustering algorithms that have been developed based on different techniques [3] [4], such as centroid based, connectivity based, density based and graph based algorithms. Each algorithm has advantages and disadvantages, and its effectiveness depends on its fit to the actual datasets that it is applied on. For example, a clustering algorithm can perform extremely well on one dataset, but gives a poor result for another dataset that presents a different type of data distribution. Among the various clustering algorithms, K-means and Agglomerative hierarchical clustering are the most popular clustering algorithms and are implemented in many big data machine learning frameworks [5], due to their simplicity in algorithm implementation and effectiveness in many practical clustering problems [4] [6].

In this paper, we propose a novel Split-merge-evolve clustering algorithm. It is similar to K-means and Agglomerative hierarchical algorithms, and is therefore very flexible in that it can be designed with various techniques, such as centroid based and linkage based algorithm implementations. However, our proposed algorithm also supports continuous cluster improvement based on user-defined clustering quality measures, which sets it apart from K-means and Agglomerative hierarchical clustering algorithms. In our experiments, we also show that it performs as well or better than the classical clustering algorithms.

The contribution of this paper can be summarized as follows:

- A novel Split-merge-evolve clustering algorithm, which is as simple and flexible as K-means and Agglomerative hierarchical algorithms in its implementation.
- Proved effectiveness of the new algorithm, shown by comparing its results with both K-means and Agglomerative hierarchical algorithms on both artificial and real life datasets.

The remainder of this paper is organised as follows.

Section 2 introduces K-means, Bisecting k-means and Agglomerative hierarchical clustering algorithms, as they are related to the Split-merge-evolve algorithm design and will also be used for benchmarking against, in the experimental results section. Section 3 details the Split-merge-evolve algorithm design with two different techniques (centroid and connectivity based) for the implementation. Section 4 explains our evaluation method and results on four different datasets. Section 5 discusses some related work, while Section 6 details the conclusion about the Split-merge-evolve algorithm and our future work.

## 2. Clustering techniques

### 2.1. K-means clustering

K-means clustering is one of the simplest and most widely used partitioning clustering algorithms [4]. Clusters are represented by a central vector of so-called centroids, which is the mean of data vectors within the clusters.

The algorithm (Algorithm 1) starts by randomly choosing  $k$  points among the data points as the initial centroids. Each point is then assigned to the closest centroid based on a particular distance measure chosen. Once the clusters are formed, the centroids for each cluster are updated. The algorithm then iteratively repeats the last two steps until the centroids do not change.

Select  $k$  points as initial centroids;

**repeat**

    Calculate distances between each centroid and each point;

    Form  $k$  clusters by assigning each point to its closest centroid;

    Recompute centroids of each clusters;

**until** centroids do not change;

**Algorithm 1:** K-means Algorithm

The key metric of this clustering algorithm resides in how to calculate the distance between a centroid and a data vector, and the centroids among data vectors. There are different kinds of distance metrics that can be used here, such as Manhattan distance, Euclidean distance. The mean value of the data vectors is generally used for the centroid value.

### 2.2. Bisecting k-means clustering

Bisecting k-means is a combination of K-means and divisive hierarchical clustering method [3] [4]. Unlike the K-means algorithm, it does not need to initialize  $k$  number of centroids before the algorithm starts.

The algorithm (Algorithm 2) starts with one cluster which contains all the data vectors, then iteratively finds a cluster and splits it in 2 parts using K-means until  $k$  numbers of clusters is reached.

The key metric of this clustering algorithm resides in how to measure the cluster quality to choose a cluster for

Assign all data as a single cluster;

**repeat**

    Compute qualities for each cluster;

    Find the worst cluster and split it into 2

        sub-clusters (using K-means cluster);

**until**  $k$  number of clusters are reached;

**Algorithm 2:** Bisecting K-means Algorithm

splitting. Within cluster average/median distance, etc. can be used for this cluster quality measure.

### 2.3. Agglomerative hierarchical clustering

Agglomerative hierarchical clustering [4] starts by taking singleton clusters which contain only one single data vector per cluster in the beginning, and continues merging two clusters at a time to build a hierarchy of clusters. This clustering technique has the advantage that it allows for cutting the hierarchy at any given level and obtaining the  $k$  number of clusters correspondingly.

Assign each data point as a new cluster

**repeat**

    Compute distances between each cluster

    Find 2 closest clusters and merge them into a new cluster

**until**  $k$  number of clusters is reached;

**Algorithm 3:** Agglomerative Hierarchical Algorithm

The key metric of this clustering algorithm resides in how to measure the distance between clusters for merging. Ward's or link based method, etc. techniques can be used for measuring distances between clusters.

## 3. Split-merge-evolve clustering

Split-merge-evolve is a novel clustering algorithm, and it is a hybrid algorithm which combines both Bisecting k-means and Agglomerative hierarchical clustering techniques. It uses *split*, *merge* and *evolve* key actions to build an iterative process to form or optimize the clustering result (Algorithm 4). The algorithm randomly divides data into  $k$  clusters initially, then repeatedly splits bad clusters and merges closest clusters to evolve the final clustering result.

The algorithm implementation is flexible. Different techniques can be used for the three key actions in the algorithm design. In the following subsections, we describe these key actions with examples of two different techniques: Centroid based and Connectivity based.

### 3.1. Split action

The split action is similar to the Bisecting k-means clustering. It is the first step in the iterative optimization process. We measure qualities of all clusters to find the worst cluster to split into a number of small clusters. The simplest way is splitting it into 2 small clusters. If we split it into

Initialize: inverse clustering quality  $e = \max$ ;

Randomly split data as  $k$  clusters

**repeat**

**split:**

    Compute qualities for each cluster

**if**  $split \neq merged$  **then**

      Find the worst cluster

**end**

**else**

      Randomly select one from top  $n$   
       ( $n \leq k$ ) largest clusters as the worst  
       cluster

**end**

    Split the worst cluster into 2 sub-clusters

**end;**

**merge:**

    Compute distances between each cluster

**if**  $split \neq merged$  **then**

      Find 2 closest clusters and merge them  
       into a new cluster

**end**

**else**

      Randomly select one from top  $n$   
       ( $n \leq k$ ) smallest clusters and merge  
       with its closest cluster

**end**

**end;**

**evolve:**

    Recompute a new quality score:  $e_{new}$

**if**  $e_{new} < e$  **then**

      update  $e_{new}$  as  $e$

      update the current clustering result as the  
       final result.

**end**

**end;**

**until** convergence criterion is reached;

**Algorithm 4:** Split-merge-evolve Algorithm

more than 2 clusters, we just need merge that same amount of clusters in the following merge step to keep the total of  $k$  clusters unchanged in each optimization iteration.

For a centroid based algorithm, we find the worst cluster based on the measure of the sum of squares of data points within the cluster to the centroid. The quality of each cluster can be calculated as the following:

$$\sum_{x \in s} \|x - c\|^2$$

where  $c$  is the centroid of a cluster  $s$ . Finally, similar to Bisecting k-means clustering, K-means is used for splitting the worst cluster.

Alternatively, we can use the average distance within cluster for a connectivity based algorithm design. The cluster quality is represented as the pairwise distance between data points within a cluster:

$$\frac{1}{|s|(|s| - 1)} \sum_{x, x' \in s} \|x - x'\|$$

after which we use Agglomerative hierarchical clustering to split the worst cluster for the average distance based approach.

For some cases during the iterative process, the merged clusters may be the same as the split clusters of a previous step after a certain number of iterations. I.e.  $split == merged$ . For example, cluster\_2 is split into 2 small clusters, cluster\_2 and cluster\_8, but it merges cluster\_2 and \_8 back to cluster\_2 again in a following merge step. It means that the final cluster result becomes the same before the splitting and merging steps. This makes all subsequent optimization iterations become worthless. Nevertheless, this problem did not very frequently occur in our experiments for the above centroid based algorithm. The reason is that K-means results are strongly affected by randomly chosen initial centroids. This randomness factor makes the results of split steps likely different each time even when splitting the same cluster. To further avoid this problem even when K-means is not used for splitting, we applied a different strategy to find the worst cluster in our algorithm design. We randomly selected one of the top (e.g. top 3) largest clusters as the worst cluster when this problem happened during the iterative process.

### 3.2. Merge action

The merge step is similar the agglomerative hierarchical clustering. We find the 2 clusters having the closest distance between them and merge them together.

For a centroid based algorithm, we use distances between centroids to represent the distance between clusters. The between cluster distance can be calculated by the centroid linkage,

$$\|c_i - c_j\|$$

where  $c_i$  and  $c_j$  are the centroids of clusters  $s_i$  and  $s_j$ , respectively.

Alternatively, we can use the average of distances between all pairs of vectors in cluster  $S_i$  and cluster  $S_j$  to represent between cluster distances as the following,

$$\frac{1}{|s_i||s_j|} \sum_{x_i \in s_i} \sum_{x_j \in s_j} \|x_i - x_j\|$$

Finally, the 2 closest distance clusters will be merged in the merge step. Similar to the split step, to further avoid  $split == merged$  iterations, we use a different strategy to find merge targets. We find the smallest cluster and its closest cluster to merge together.

### 3.3. Evolve action

We consider each combined split, merge and evolve action as one complete optimization step in the iterative process. The evolve action is used to bring the final clustering result towards the user chosen or defined objective, i.e. maximize the defined clustering quality measures or minimize the inverse clustering quality. We show two simple quality measures used in our implementation for general

cases. Choosing or defining the most suitable clustering quality measures may require additional analysis for different practical clustering problems [7].

For a centroid based algorithm, we use average distances of data vectors to their closest centroids as a measure of the clustering quality:

$$\frac{1}{|s|} \sum_{x \in s} \|x - c\|^2$$

Alternatively, we could use the internal cluster evaluation for measuring the clustering quality for connectivity based design. The smaller distance of data vectors within cluster with a longer distance of data vectors between clusters will give a better clustering quality. It can be calculated as the following:

$$\frac{\min_{x_i \in s_i, x_j \in s_j} \|x_i - x_j\|}{\max_{x, x' \in s} \|x - x'\|}$$

When we have a better clustering quality or optimization score, we will evolve the final clustering result.

### 3.4. Convergence criterion

The iterative process containing the split, merge and evolve steps will be terminated when a convergence criterion is reached. The process is meant to optimize or minimize inverse clustering quality score which is based on user defined clustering quality or internal clustering evaluation metrics. This allows users to specify an acceptable clustering quality score to terminate the process when the inverse clustering quality falls below the acceptable value. In this paper, our experiment simply defines a maximum number of iterations to terminate the process.

### 3.5. Graphical comparison

Most existing clustering algorithms including the Split-merge-evolve algorithm require an iterative process over the dataset to achieve convergence. Nevertheless, the Split-merge-evolve algorithm is unique to existing commonly used clustering algorithms, such as the ones we discussed. To further show the differences, we use table 1 to give a graphical comparison.

Different algorithms start with different initial clusters and then use different iterative steps and techniques to achieve the final result (below). Also, their iterative steps focus on different key metrics.

- K-means clustering forms  $k$  clusters initially, then relocates data points between different clusters iteratively.
- Bisecting k-means clustering starts with a single cluster initially, then splits a cluster iteratively.
- Agglomerative hierarchical clustering starts with each point as a cluster initially, then merges two clusters iteratively.

- Split-merge-evolve clustering forms  $k$  cluster initially, then splits and merges selected clusters iteratively.

The Split-merge-evolve clustering algorithm, same as K-means, can maintain  $k$  number of clusters over the iterative optimization process. It gradually improves the quality of the formed  $k$  clusters during the process. The algorithm can be stopped in the middle of the process and still provide a  $k$  clusters output. As a result, Split-merge-evolve algorithm has the flexibility to trade off between computation time and accuracy, compared to Bisecting k-means and Agglomerative hierarchical. This also makes the algorithm have the ability to optimize the clustering result in some scenarios where the data samples are dynamically changing. For example, new data samples may be added in to existing clusters. On the other hand, Split-merge-evolve does not show cluster structure in an informative tree diagram.

An important key metric during the clustering process of Split-merge-evolve algorithm is a user chosen or defined clustering quality metric or internal evaluation. The algorithm evolves the clustering result towards the user expected high quality result, although there is no ground truth or labelled data involved during the clustering. It gives the advantages of allowing users to specify what they believe is correct and make the clustering process towards that direction.

## 4. Algorithm evaluation

To show comparable results in our evaluation, we compare the K-means, Agglomerative hierarchical and Split-merge-evolve clustering with different labelled datasets: 2 artificial datasets (Dataset 1 and 2) and 2 real life datasets (Dataset 3 and 4) respectively. To avoid data privacy issues and also to have reproducible results for others, our evaluation in this paper shown are based on open real life datasets.

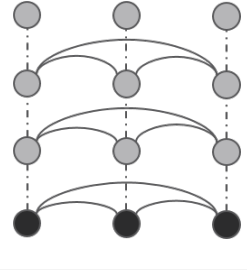
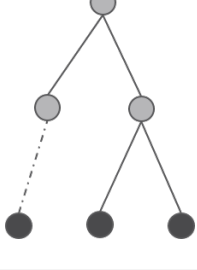
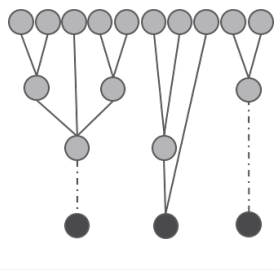
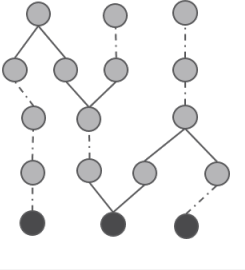
Adjusted Rand Index (ARI) [8] [9] is one of standard clustering evaluation measurements and we also use it in our evaluation. It measures the similarity between the clustering algorithm output  $P$  and clustering ground truth  $G$ . The higher the ARI score (max value = 1) of a clustering algorithm for a dataset, the more effective of the algorithm is for the dataset. It can be calculated as:

$$ARI(P, G) = \frac{2(ab - cd)}{(a + c)(c + b) + (a + d)(d + b)}$$

where  $a$  is for data vectors in the same class both in  $P$  and  $G$ ,  $b$  is data vectors in the same class in  $P$ , but different classes in  $G$ .  $c$  is data vectors in different classes in  $P$ , but in the same class in  $G$ .  $d$  is data vectors in different classes both in  $P$  and  $G$ .

We use the same distance technique for centroid/connective based Split-merge-evolve algorithm, K-means and Agglomerative hierarchical algorithm respectively. Euclidean distance is used to measure the distance to centroids or between data vectors. For every dataset we run each algorithm 20 times and take an average ARI

TABLE 1: Graphical comparison of different algorithms

	K-means	Bisecting k-means	Agglomerative hierarchical	Split-merge-evolve
Initial cluster	$k$ clusters	1 cluster	$n$ clusters	$k$ clusters
Iterative process				
Key metric	distance to centroids	within cluster distance	between cluster distance	+ clustering quality/ internal evaluation

score as the final measurement. We normalized all attributes of datasets before clustering such that the means of all attributes are the same for each dataset, contributions of the features to distance measures are approximately equalised [10].

#### 4.1. Dataset 1

Dataset 1 is a simple artificial two dimensional dataset which has two classes. It has a total of 350 data vectors that are clearly separated in two classes without any overlapping (Figure 1). We randomly generate 150 data points for each class firstly, and then add additional 50 noisy data points towards the first class. The x and y ranges for the two classes are as follows:

class **A**:  $x=[0.3, 0.5], y=[0.3, 0.5]$   
class **A** noises:  $x=[0.1, 0.5], y=[0.1, 0.5]$   
class **B**:  $x=[0.8, 1.0], y=[0.8, 1.0]$

Figure 1: Dataset 1

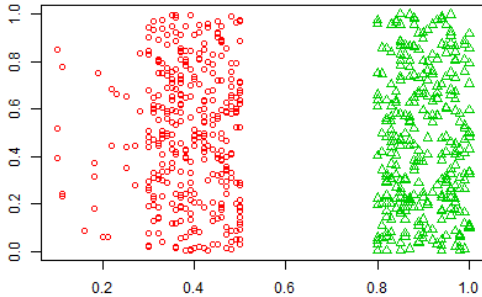


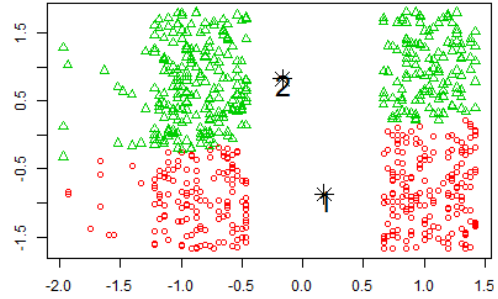
Table 2 shows the final result. Both connectivity based Split-merge-evolve and Agglomerative hierarchical algorithms give a perfect result which is the max ARI score. The centroid based Split-merge-evolve algorithm follows closely. The K-means algorithm does not perform very well comparing to others.

Since this dataset has a clear space between two classes, we were expecting all clustering algorithms to give perfect results for this dataset. The reason why K-means has a much

TABLE 2: Result of dataset 1

Algorithm	ARI
K-means	0.512
Agglomerative hierarchical	<b>1.000</b>
Split-merge-evolve (centroid based)	0.946
Split-merge-evolve (connectivity based)	<b>1.000</b>

Figure 2: Case of bad output from K-means



lower score comparing to the other algorithms is because it randomly initializes centroids, which strongly affects the algorithm performance. For some cases, badly initialized centroids could result in a low quality result in K-means (Figure 2 shows an example). The centroid based Split-merge-evolve may encounter the same problem during the iterative process in the split step since it uses K-means for the split step, however it can fix the problem in subsequent optimization iterations in most cases. Hence, the centroid based Split-merge-evolve algorithm gives a much higher score comparing to K-means.

To better understand the internal process steps of Split-merge-evolve algorithm, we use Figure 3 to show the cluster formation during the iterative process of the centroid based Split-merge-evolve algorithm. At the initial stage, the algorithm randomly divides the data into 2 clusters (Figure 3.a). It splits the cluster 1 into cluster 1 and 3, then merges cluster 1 and 2 as cluster 1 in the first iteration. At this stage (figure 3.c), we can see that the cluster 1 has all the data vectors of

its own class (data of the right partition) but also has some data from the left partition. In the second iteration, the data from the left partition of cluster 1 is split out and merged into cluster 2. The correct cluster result is formed at the end of the second iteration (figure 3.e). Although we had total 30 optimization iterations in this case, the final result would not be updated in subsequent iterations.

#### 4.2. Dataset 2

Dataset 2 is another two dimensional artificial dataset with three classes. Unlike dataset 1, classes do not have clear boundaries between them in this dataset, Figure 4. Data points are spread in a normal distribution within each class. The mean and standard deviation (sd) of x and y in each class is as follows:

class A:  $x=[\text{mean}=2, \text{sd}=1.5], y=[\text{mean}=2, \text{sd}=1.5]$   
class B:  $x=[\text{mean}=5, \text{sd}=1.5], y=[\text{mean}=5, \text{sd}=1.5]$   
class C:  $x=[\text{mean}=1, \text{sd}=1.5], y=[\text{mean}=6, \text{sd}=1.5]$

TABLE 3: Result of dataset 2

Algorithm	ARI
K-means	0.859
Agglomerative hierarchical	0.360
Split-merge-evolve (centroid based)	<b>0.887</b>
Split-merge-evolve (connectivity based)	0.886

Table 3 shows the final result. The centroid based Split-merge-update algorithm has the best performance followed by the connectivity based Split-merge-evolve. The Agglomerative hierarchical algorithm does not perform well showing a very low score in this case.

#### 4.3. Dataset 3

Dataset 3 is a real life dataset which is used to quantify the morphologic variation of Iris flowers for three related species [11]. This classic dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features (Sepal length, Sepal width, Petal length, Petal width) were measured from each sample.

TABLE 4: Result of dataset 3

Algorithm	ARI
K-means	0.566
Agglomerative hierarchical	0.562
Split-merge-evolve (centroid based)	<b>0.620</b>
Split-merge-evolve (connectivity based)	0.559

Table 4 shows the final result. The centroid based Split-merge-evolve algorithm has the best performance. The rest of the algorithms have results that follow very close.

Figure 5 shows examples of iterative optimization steps of the centroid based Split-merge-evolve algorithm on the

Figure 3: Iterative process of Split-merge-evolve on dataset1

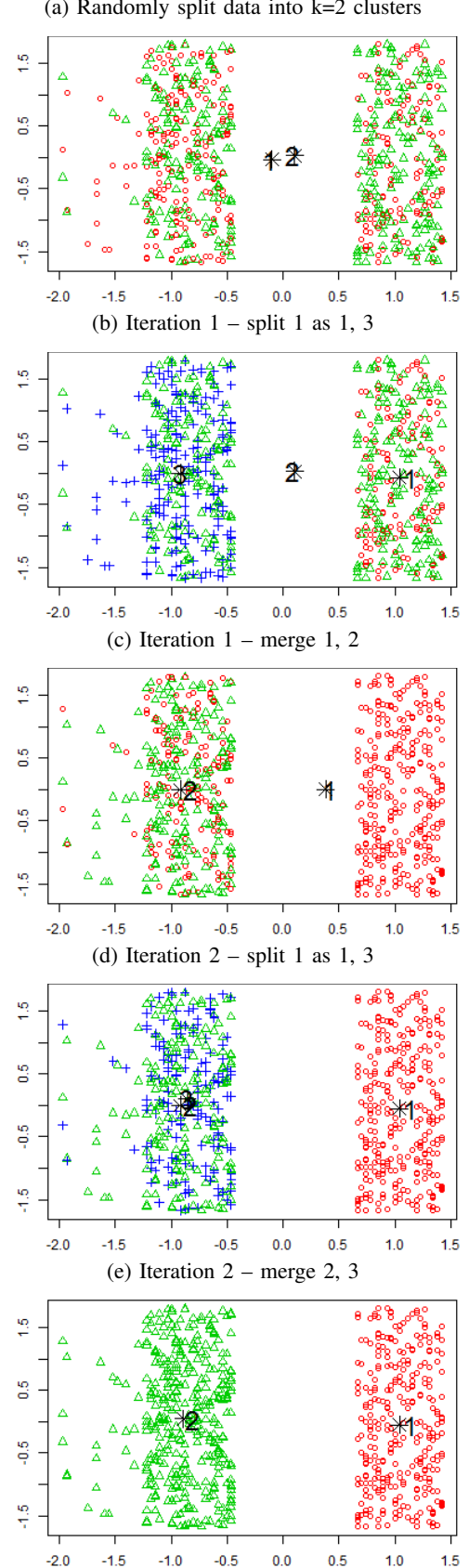
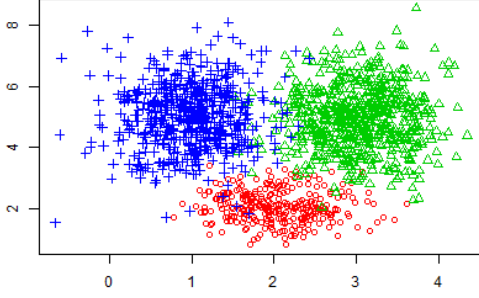


Figure 4: Dataset 2



dataset. The figure only shows the first two iterative steps due to space limitations. Nevertheless, we can clearly see that cluster 2 is already formed after the first two iterative steps.

#### 4.4. Dataset 4

Dataset 4 is another real life dataset from USA forensic science service [12]. It contains 6 types of glasses defined in terms of their oxide content and 9 attributes of glass information that are collected for 214 glasses.

TABLE 5: Result of dataset 4

Algorithm	ARI
K-means	0.174
Agglomerative hierarchical	0.019
Split-merge-update (centroid based)	<b>0.185</b>
Split-merge-update (connectivity based)	0.019

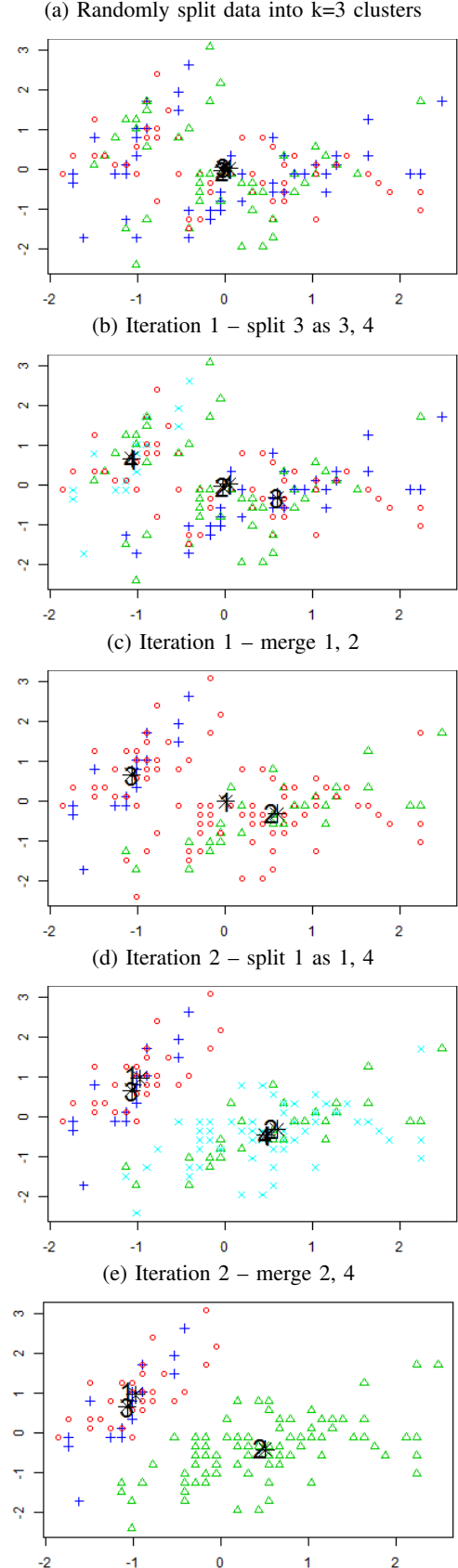
Table 5 shows the final result. The Split-merge-evolve algorithm has the best result, which is slightly better than K-means, although all algorithms give low ARI scores for this dataset.

## 5. Related work

As we presented in Section 3.5, clustering algorithm design can be compared and contrasted in two main aspects: key metrics and iterative process. In this section, we discuss related work in relation to these two aspects.

For the key metrics aspect, common clustering algorithms are mainly focused on metrics of distance and similarity measures, such as commonly used Minkowski distance, Euclidean distance and Cosine similarity. These metrics can be used to measure distance to centroids, within cluster distances, etc. There are also many algorithms designed with more advanced or complex distance and similarity measures, e.g. ultra-metric distance [13], weighted distance [14], etc. In contrast, our approach introduces user defined or chosen clustering quality metrics into the cluster algorithm design. The clustering objective is to achieve good user defined clustering quality, rather than only focusing on distance and similarity measures.

Figure 5: Iterative process of Split-merge-update on dataset3





From the iterative process point of view, common clustering algorithms involve *relocating data points* (e.g. K-means), *splitting clusters* (e.g. Bisecting k-means), *merging clusters* (Agglomerative hierarchical), etc. basic actions in the iterative process design. Many hybrid algorithms [15] [16] [17] [18] [19] may combine more than one of these basic actions in their algorithm design including our Split-merge-evolve algorithm. Although many algorithms may use the same basic actions in their algorithm design, however their iterative process design can be very different. For example, [15] [18] defined a set of splitting and merging action conditions, where optional splitting and merging actions will only be triggered during the iterative process when the conditions are met. [20] firstly identifying groups of vectors which are sufficiently close to each other and distant from all others, that they should never be separated. It treats these groups as individual objects then applies splitting or merging actions to the formal final clusters. [19] uses a splitting action initially to search for the best possible  $k$  number of clusters, then merges data vectors into desired  $k$  clusters. [21] and [22] split data into a large amount of small clusters firstly, then merge these small clusters based on their centroids or patterns into desired  $k$  clusters. In contrast, our approach does not define any thresholds or conditions for splitting and merging actions. A combined splitting and merging action as a single step is used in our iterative process design, and  $k$  number of clusters is maintained during the whole process. Additionally, we have the evolve action which is a selective process to evolve the final cluster result during the iterative process.

## 6. Conclusion

Clustering is an important problem in big data analytic. Our proposed Split-merge-evolve algorithm is a simple, flexible  $k$  clustering algorithm. The algorithm randomly divides data into  $k$  clusters initially, then repeatedly splits bad clusters and merges the closest clusters to evolve the final clustering result. Various techniques can be applied in the algorithm implementation, such as centroid and connectivity based, as we presented in the paper.

The Split-merge-evolve algorithm is unique comparing to common K-means and Agglomerative hierarchical from the point of view of the iterative process and the key metrics, as explained in Section 3 and 5. An important key metric during the clustering process of Split-merge-evolve algorithm is a user chosen or defined clustering quality metric or internal evaluation. The algorithm evolves the clustering result towards to the user expected high quality result. Since the algorithm maintains  $k$  numbers of clusters for the process. It makes the algorithm have the ability to optimize the clustering results in some scenarios where the data samples are dynamically changing. In our experiments of four datasets including two real life datasets, the Split-merge-evolve algorithm shows its effectiveness by giving better results comparing to both most commonly used K-means and Agglomerative hierarchical algorithms. Our further work will involve to evaluate the algorithm with larger

scale and different types of datasets, including dynamic dataset in streaming environment.

## References

- [1] M. Wang and S. Handurukande, "A streaming data anomaly detection analytic engine for mobile network management," in *IEEE International Conference on Cloud and Big Data Computing*, 2016.
- [2] K. Julisch, "Mining alarm clusters to improve alarm handling efficiency," in *Computer Security Applications Conference*, 2001.
- [3] C. C. Aggarwal, "An introduction to cluster analysis," in *Data Clustering Algorithms and Applications*. Chapman and Hall/CRC, 2014, pp. 1–28.
- [4] C. K. Reddy and B. Vinzamuri, "A survey of partition and hierarchical clustering algorithms," in *Data Clustering Algorithms and Applications*. Chapman and Hall/CRC, 2014, pp. 87–106.
- [5] C. Jin, R. Liu, and Z. Chen, "A scalable hierarchical clustering algorithm using spark," in *IEEE First International Conference on Big Data Computing Service and Applications*, 2015.
- [6] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of common document clustering techniques," in *KDD Workshop on Text Mining*, 2000.
- [7] C. Tomasini, L. Emmendorfer, E. N. Borges, and K. Machado, "A methodology for selecting the most suitable cluster validation internal indices," in *ACM Symposium on Applied Computing*, 2016.
- [8] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.
- [9] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *International Conference Artificial Neural Networks*, 2009.
- [10] M. M. Suarez-Alvarez, D.-T. Pham, M. Y. Prostov, and Y. I. Prostov, "Statistical approach to normalization of feature vectors and clustering of mixed datasets," *The Royal Society A Mathematical Physical and Engineering Sciences*, vol. 468, no. 2145, pp. 2630–2651, 2012.
- [11] Iris dataset. <http://archive.ics.uci.edu/ml/datasets/Iris>.
- [12] Glass identification dataset - us forensic science service. <http://archive.ics.uci.edu/ml/datasets/Glass+Identification>.
- [13] L. Zheng, T. Li, and C. Ding, "Hierarchical ensemble clustering," in *IEEE International Conference on Data Mining*, 2010.
- [14] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, "Automated variable weighting in k-means type clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 77, no. 5, pp. 657–668, 2005.
- [15] E. Lughofer, "Dynamic evolving cluster models using on-line split-and-merge operations," in *International Conference on Machine Learning and Applications*, 2011.
- [16] K. Murugesan and J. Zhang, "Hybrid hierarchical clustering: an experimental analysis," in *University of Kentucky, Lexington. Technical Report*, 2011, p. 2011.
- [17] D. Chaudhuri, B. Chaudhuri, and C. Murthy, "A new split-and-merge clustering technique," *Pattern Recognition Letters*, vol. 13, no. 6, pp. 399–409, 1992.
- [18] R. Fa and A. K. Nandi, "Smart: Novel self splitting-merging clustering algorithm," in *European Signal Processing Conference*, 2012.
- [19] J. Senthilnath, D. Kumar, J. Benediktsson, and X. Zhang, "A novel hierarchical clustering technique based on splitting and merging," *International Journal of Image and Data Fusion*, vol. 7, no. 1, pp. 19–41, 2016.
- [20] H. Chipman and R. Tibshirani, "Hybrid hierarchical clustering with applications to microarray data," *Biostatistics*, vol. 7, no. 2, pp. 286–301, 2006.
- [21] R. Chitta and M. N. Murty, "Two-level k-means clustering algorithm for k-t relationship establishment and linear-time classification," *Pattern Recognition*, vol. 43, no. 3, pp. 796–804, 2010.
- [22] T. H. Sarma, P. Viswanath, and B. E. Reddy, "A hybrid approach to speed-up the k-means clustering method," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 2, p. 107117, 2013.