COMPARISON OF FOUR ALGORITHMS FOR ONLINE CLUSTERING

Xinchun Yang[1,2], Wassim Kabbara[1,3]

[1]Department of Computer Engineering, Centrale Supélec, Paris, France
[2]Department of Electrical Engineering, Tsinghua University, Beijing, China
[3]Department of Electrical Engineering and Electronics, Lebanese University, Tripoli, Lebanon

ABSTRACT

This paper concludes and analyses four widely-used algorithms in the field of online clustering: sequential K-means, basic sequential algorithmic scheme, online inverse weighted K-means and online K-harmonic means. All algorithms are applied to the same set of self-generated data in 2-dimension plane with and without noise separately. The performance of different algorithms is compared by means of velocity, accuracy, purity, and robustness. Results show that the basic sequential K-means online performs better on data without noise, and the K-harmonic means online performs is the best choice when noise interferes with the data.

KEYWORDS

Sequential Clustering, online clustering, K-means, time-series clustering

1. INTRODUCTION TO SEQUENTIAL CLUSTERING

Clustering is the process of grouping a set of objects according to certain criteria such that members in each group are similar. It is one of the most important tools in fields of machine learning and data mining, and is widely used in areas like social media analysis, image processing, psychological analysis, etc. Traditional algorithms have been well-established, but most of them aim at still and unchanged data. With the frequent appearance of big data and the mutative market demand, dynamic and time-series data are usually preferred, and the algorithms couldn't meet the requirement of users in many cases any more. Sequential clustering (or online clustering), as the name indicates, is the kind of clustering that deal with sequential. Its rapidity and accuracy on processing time-sequenced data in real-time applications make it the perfect solution for the uprising problems. Therefore, research into it is becoming really meaningful.

Currently, lots of studies have been conducted into the sequential clustering. Many, if not overmuch, theories are proposed and algorithms are developed, bringing prosperity along with inconvenience to this field. Software engineers and program developers sometimes may get confused when trying to select the algorithm for their work. A direct comparison between algorithms should be conducted, so that performances and characteristics of different algorithms can be listed clearly as important references for users. This paper is aiming at comparing and analysing the performances of different prevailing algorithms on sequential clustering.

To do the task, three parts of work are required: to explain the basic theory carefully, to apply different algorithms to the same problem, and to analyse and evaluate the result in a convincing

Natarajan Meghanathan et al. (Eds) : DaKM, SIPP, CCSIT, NCWMC - 2018
pp. 01–19, 2018. © CS & IT-CSCP 2018 DOI : 10.5121/csit.2018.81501

way. Many researches have done great in some aspects, yet few have completed all the three parts. Nevertheless, this paper is inspired largely by their works.

Aghabozorgiet al. [1]did a really outstanding work on reviewing the history of sequential clustering and categorizing different algorithms according to different criteria. He concluded that there are basically 3 categories of all the algorithms: partitioning algorithm, which create k groups from n unlabelled objects in the way that each group contains at least one object; hierarchical algorithm, which produce a hierarchy of clusters using agglomerative or divisive algorithms; multiple-step algorithm, which combines different methods by dividing the work into multiple steps. Barbakhet al. [2] wrote a comprehensive book on clustering algorithms and explained the mechanisms thoroughly, including that of DBSCAN, IEK, IWKO, KHMO, etc, which has been a good reference for this paper. Their works view the field of sequential clustering from the top, providing good understandings and great perspectives, but lack in the stage of operation. Sardar et al.[3] modified traditional K-means algorithm into parallel one so that it can be implemented on top of Hadoop with increased accuracy and efficiency; Huang et al.[4] developed a new time-series K-means algorithm, which would improve the performance on exploiting inherent subspace information of a time series data set; Yang et al.[5] constructed a new framework combining the advantages of clustering and classification, and compared the result with traditional frameworks. Zhao et al. [6] developed an algorithm for mixed data based on information entropy, and test the data with 8 different datasets under 3 evaluation measures. Though they managed to improve one certain algorithm of clustering rather than compare different algorithms, their studies are really helpful on the methodologies of implementing and evaluating their algorithms. There are also more studies explaining the details and pros and cons for a single algorithm, such as the book by manning explaining everything about sequential K-means[7] and the report by Grzegorzek presenting the basic sequential algorithm scheme [8]. They are not helpful in their methodology and study structure, but they are very good teachers.

Inspired by the previous works, this paper chooses four algorithms to study: sequential K-means, BSAS, IWKO and KHMO, as they are based on the similar idea of partitioning, and they can be easily implemented on self-generated data with existing tools. Theories will be explained in the next part.

The format of remaining paper is that: the Section 2 describes the theories and characteristics of four most widely-used partitioning clustering methods, the Section 3presents the implementation of those algorithms on a certain set of data, and the Section 4 evaluates their effectiveness and performances to decide on the best algorithm among the four.

2. THEORIES OF FOUR COMMONLY-USED ALGORITHMS FOR SEQUENTIAL CLUSTERING

2.1. Sequential K-means Algorithm

The first algorithm we are discussing is the Sequential K-means Algorithm. The normal offline K-means algorithm[5] start with K randomly chosen centers(or prototypes). All the data are clustered by their Euclidean distance to the center and form K clusters. Calculate the mean value for data in each cluster, set the mean values as new centers and then go through the same process, getting K new clusters with new centers. Repeat the process until it stabilizes so that a good set of clusters can be decided.

Instead of having the examples all at once in the beginning and do the clustering afterwards, the sequential algorithm updates one example at a time, cluster the new example and re-calculate the center for this particular cluster. [6] Awidely used algorithm is as follow.

Make initial guesses for the means $m_1, m_2 \dots, m_k$*;*
Set the counts $n_1, n_2 \dots, n_k$ *to zero;*
Until interrupted:
    Acquire the next example, x*;*
    If m_i *is closest to x:*
        Increase n_i *for 1;*
        Replace m_i *by* $m_i + \frac{x - m_i}{n_i}$.

This method is accurate but involves unnecessary calculations. A similar algorithm replaces the $\frac{1}{n_i}$ part by a consistent learning rate α between 0 and 1, which sacrifices the relative accuracy for a higher speed.

Define a constant α *between 0 and 1;*
Make initial guesses for the means $m_1, m_2 \dots, m_k$*;*
Until interrupted:
    Acquire the next example, x;
    If m_i *is closest to x, replace* m_i *by* $m_i + \alpha(x - m_i)$.

This algorithm has a characteristic of being 'forgetful'. A newer example would have a higher weight on calculating new clusters than the old ones, as the final value of m_i can be represented as

$$m_i = (1-\alpha)^i m_0 + \alpha \sum_{k=1}^{i} (1-\alpha)^{n-k} x_k$$

where m_0 is the initial guess, and x_j is the j^{th} of n examples used to form m.

Sequential data can be processed more quickly and efficiently with the Sequential K-means Algorithm, but a question on this algorithms is how to choose the initial prototypes. A good set of initial value could vastly reduce the amount of calculation, while a bad set would do the opposite.

2.2. Basic Sequential Algorithmic Scheme (BSAS)

In the sequential K-means algorithms we've just discussed, the number of clusters is pre-determined, but the number along with many other details of the upcoming vectors are not known a priori in many circumstances, making the former algorithm useless. To avoid this problem, the BSAS is proposed.

The method of BSAS obeys two certain rules: All vectors are presented to the algorithm only once; the clusters are gradually generated in the clustering process. [3]When the distance between an upcoming example and any other clusters is beyond a threshold, a new cluster is created. The mechanism of this algorithm is stated below.

Define the number of clusters m and its roof limit q;
Initialize m = 1;
Define the first cluster $C_m = \{x_1\}$*;*
For i = 2 to N:
    Find $C_k: d(x_i, C_k) = \min_{1 \le j \le m} d(x_i, C_j)$
    If $d(x_i, C_k) > \theta$
        m = m + 1;
        $C_m = x_i$*;*
    Else:

$C_k = C_k \cup x_i$;
Update representatives if necessary.

The representative is used to calculate the distance between examples and clusters, and it is usually the mean value of all vectors in a single cluster. It is updated by the following equation:

$$m_{c_k}^{new} = \frac{(n_{c_k}^{new} - 1)m_{c_k}^{old} + x}{n_{c_k}^{new}}$$

Sometimes people also use the distance between the new example and the nearest or farthest vector in a cluster as representative, but the mean vector representative outstrips these by its accuracy and concision.

Problem of this algorithm is that the result of clustering is severely influenced by the order of coming examples, and an improved BSAS algorithm introduces two threshold values to solve this problem. [7] One threshold is used to decide whether to create a new cluster as before, while the other – bigger than the first one – is used to decide whether to put a new example into an existed cluster. The detailed algorithm is stated below.

Define the number of clusters m and initialize m = 1;
Define the size of store j and initialize j = 1;
Define two threshold values θ_1 *and* θ_2 *where* $\theta_1 > \theta_2$;
Define the first cluster $C_m = \{x_1\}$;
For i = 2 to N:
    Find C_k: $d(x_i, C_k) = \min_{1 \le j \le m} d(x_i, C_j)$;
    If $d(x_i, C_k) > \theta_1$:
        $m = m + 1$;
        $C_m = x_i$;
    Else if $d(x_i, C_k) < \theta_2$:
        $C_k = C_k \cup x_i$;
        Update representatives if necessary;
    Else:
        Store x_i;
        $j = j + 1$;
While j ≠0:
    For i = 2 to j:
        Repeat the former loop but get example from the store;
    Randomly select an example from the store and create a new cluster.

This method involves a great larger amount of calculation, but it prevents the effect of coming data's order in the clustering process. It is also important to note that this modification to the BSAS will make it work better by only updating the prototypes with the close points and create new clusters with the far points, and neglecting the points that come into between and not classifying them so they will not affect negatively on the update of the prototype. In the end, those unclassified points will be reclassified into the established clusters and prototypes. However, this algorithm is not 100% online, as we cannot have the whole data set and then do the clustering. We will not discuss the implementation of the improved BSAS in this paper.

2.3. Inverse Weighted K-means Online (IWKO) Algorithm

The two methods we've discussed shared a same problem – an upcoming example would only influence one single cluster. This may have weird outcome as forming odd-looking clusters or joint clusters. The IWKO is designed to solve this problem.

To begin with, we need to define a performance function. [4]

$$J_{K_m} = \sum_{i=1}^{N} \min_{1 \le j \le K} \left\| x_i - m_j \right\|^2$$

Add an auxiliarypart considering the influence of other prototypes. Let m_k be the closest prototype to x_i,and improve the function as below,

$$J_{IWK} = \sum_{i=1}^{N} [(\sum_{i=1}^{N} \frac{1}{\left\| x_i - m_j \right\|^p}) \min_{1 \le k \le K} \|x_i - m_k\|^n]$$

Where n and p are two values indicating the weight of the optimal prototype and the other prototypes. The performance function for a single data point should be

$$J_{IWK}(x_i) = \left(\sum_{i=1}^{N} \frac{1}{\left\| x_i - m_j \right\|^p} \right) \|x_i - m_k\|^n = \|x_i - m_k\|^{n-p} + \sum_{j \ne k} \frac{\|x_i - m_k\|^n}{\left\| x_i - m_j \right\|^p}$$

In order to get the ideal clustering, we need to minimize the distance between the data point and the closet prototype, and maximize the distance between that and other prototypes. That is to say,

$$\frac{\partial J_{IWK}(x_i)}{\partial m_k} = -(n-p)(x_i - m_k)\|x_i - m_k\|^{n-p-2} - n(x_i - m_k)\|x_i - m_k\|^{n-2} \sum_{j \ne k} \frac{1}{\left\| x_i - m_j \right\|^p} = (x_i - m_k) a_{ik}$$

$$\frac{\partial J_{IWK}(x_i)}{\partial m_j} = p\left(x_i - m_j\right) \frac{\|x_i - m_k\|^n}{\left\| x_i - m_j \right\|^{p+2}} = (x_i - m_k) b_{ij}$$

should all be 0 when the clustering is perfectly done.

The IWKO operates with a similar idea as the sequential K-means, but with a slight difference of adjusting all the prototypes instead of the nearest one. The goal is to optimize the performance function, and the partial derivative of the performance function would be a perfect subtraction on the old prototypes as this would always 'ease' the difference and drives the performance function towards the optimal. We can write the algorithm below in short.

$$\Delta m_k = -\mu(x_i - m_k) a_{ik}$$

$$\Delta m_{j \ne k} = -\mu\left(x_i - m_j\right) b_{ij}$$

Where the $a_{ik}$and $b_{ij}$are defined above(choose p = -1 in order to make calculation easier), and the μ is a 'learning rate' between 0-1. The prototype k is selected by

$$\mathrm{k} = arg \min_{1 \le k^* \le K} \|x_i - m_{k^*}\|$$

and are updated as

$$m_k^{new} = m_k^{old} - \mu(x_i - m_k) a_{ik}$$

$$m_{j \ne k}^{new} = m_j^{old} - \mu(x_i - m_j) b_{ij}$$

Despite the comparative accuracy, a disadvantage of this algorithm is the complexity. It takes too much calculation for every single step, which slows the process severely.

2.4. K-Harmonic Means Online Mode Algorithm(KHMO)

The IWKO solved the problem of former algorithms, but its redundancy on calculation is a big shortcoming. The KHMO could solve the same problems with a more concise calculation but less accuracy. In this case, the performance function is defined as the harmonic average of the distances from each data point to the prototypes,

$$J_{HA} = \sum_{i=1}^{N} \frac{K}{\sum_{k=1}^{K} \frac{1}{\|x_i - m_k\|^2}}$$

Get the partial derivative:

$$\frac{\partial J_{HA}}{\partial m_k} = -K \frac{2(x_i - m_k)}{\|x_i - m_k\|^4 (\sum_{j=1}^{K} \frac{1}{\|x_i - m_j\|^2})^2}$$

And the prototypes are updated as

$$m_k^{new} = m_k^{old} + \frac{2K(x_i - m_k^{old})}{\|x_i - m_k^{old}\|^4 (\sum_{j=1}^{K} \frac{1}{\|x_i - m_j\|^2})^2}$$

This algorithm is cleaner than the IWKT by decreasing the calculation work. It also includes all prototypes other than the ideal prototype, but by using the harmonic average, it avoids dealing with the ideal prototype and others separately and makes calculation much easier.

The pros and cons of all the algorithms are listed in the chart below.

Table 1. Pros and cons for each algorithm.

Algorithms	Pros	Cons
K-means(forgetful)	More accurate than the unforgetful one, and simpler than the rest three	Slower than the unforgetful one, with an initial-prototype-choosing problem
K-means(unforgetful)	Faster then the unforgetful one, and simpler than the rest three	Not so accurate as the forgetful one with the same initial value problem
BSAS	Universal as the cluster number is not pre-defined	Result is influenced by data upcoming order, with a threshold-deciding problem
IWKO	Adjust every cluster on an upcoming example, making result more accurate	Involving huge amount of calculation which makes it really slow
KHMO	Adjust every cluster on an upcoming example, and simpler than IWKO	May be less accurate than IWKO and slower than the rest three

3. EMPIRICAL STUDY

In order to get a better view of the theories, we apply the algorithms on a set of self-generated 2-dimension data in MATLAB and analyze the results. We use a normal distribution with a σ=0.6 as the distribution of each cluster and choose the origins randomly to form 6 clusters with 1000 points. Noise is generated by a uniform distribution measuring 4% of the data set. The data with and without noise are shown below.

Most of our algorithms need initial guess for the prototypes, thus we have made it more difficult on the algorithms by considering a worst-case scenario for the initialization of the prototypes by making a random initialization located outside the gathering of the clusters. We took this decision in order to give a true insight into the performance of the algorithms without any initial advantages.

Self-generated data is shown as below. In the graphs, data are represented as stars, initial guesses of the prototypes are represented as black crosses, and final prototypes after all data have been processed are represented as blue circles.

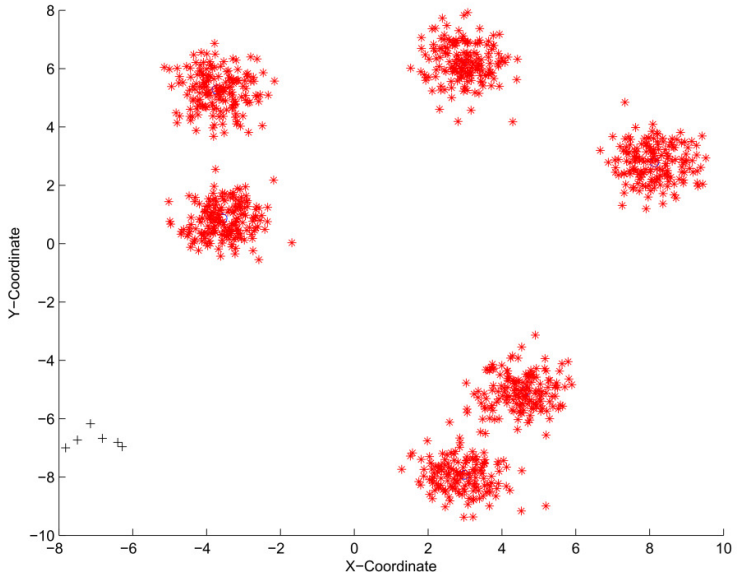

Figure 1. Data generation with 1000 scattered data around 6 means following normal distribution sigma = 0.6 of which 0% are scattered noise

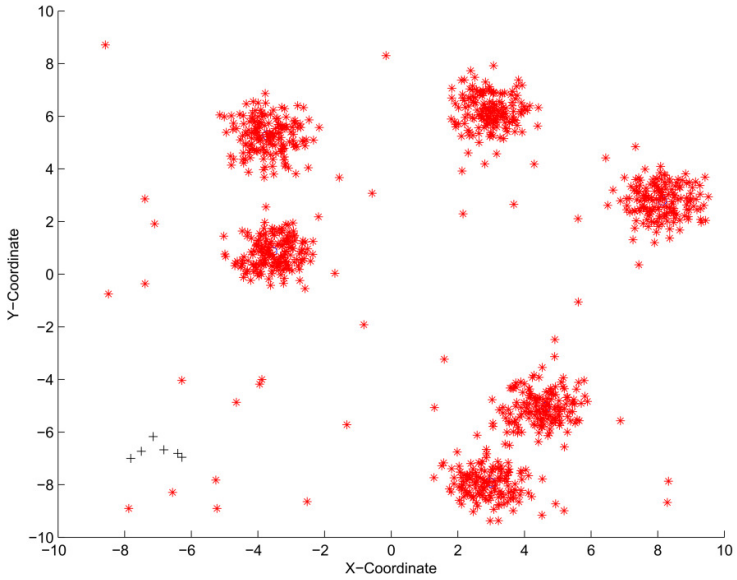

Figure 2. Data generation with 1000 scattered data around 6 means following normal distribution sigma = 0.6 of which 4% are scattered noise

In order to make performances of the algorithms comparable, we use the same sets of data for the whole study. Examples are sequentially fed to generate the time-series data. In order to mimic the sequential behavior of data, at each iteration, we will select a random point from the generated data and do the processing on it.

First, we will test the proposed algorithms on the generated data without the presence of the noise factor. Then we will introduce a uniform noise on our data and check the effect on each algorithm.

3.1. Study Without Noise

3.1.1. Implementation of Sequential K-means Algorithm

We implement the unforgetful one first. Select the initial prototypes randomly, and according to the algorithm, they should spread to approach the ideal prototypes and form the 6 clusters.

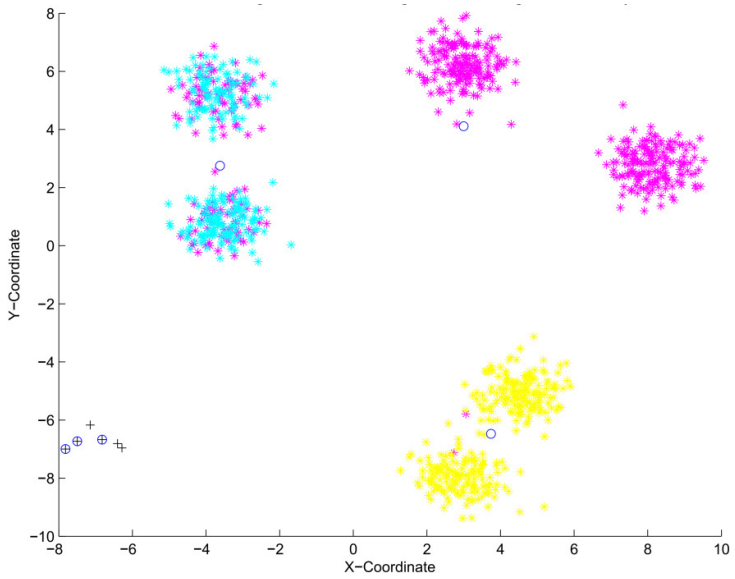

Figure 3. Classification using Online K−means unforgetful with initializing means Randomly

We see from the results that only 3 prototypes have been updated and relocated from their initial positions while the rest did not get updated. This is because that in the online K-means, the new coming point gets allocated to the clusters defined by the closest prototype to the point (code colored), and only this prototype gets updated. The major problem occurs when we initialize some prototypes in such way they will never be the closest to the data points, hence they will never be updated.

Then we run the forgetful one with different learning rates a = 0.1 and a = 0.75 separately.

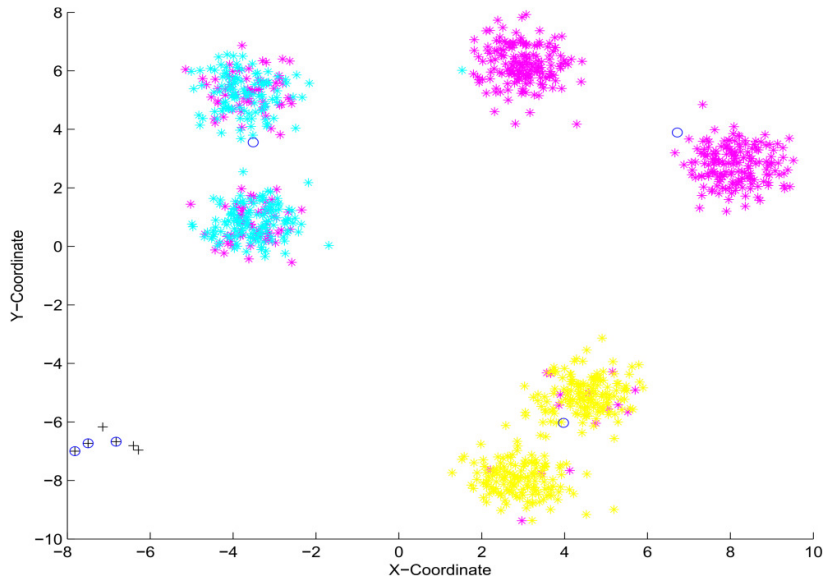

Figure 4. Classification using Online K−means forgetful (learning coefficient α=0.1) with initializing means randomly

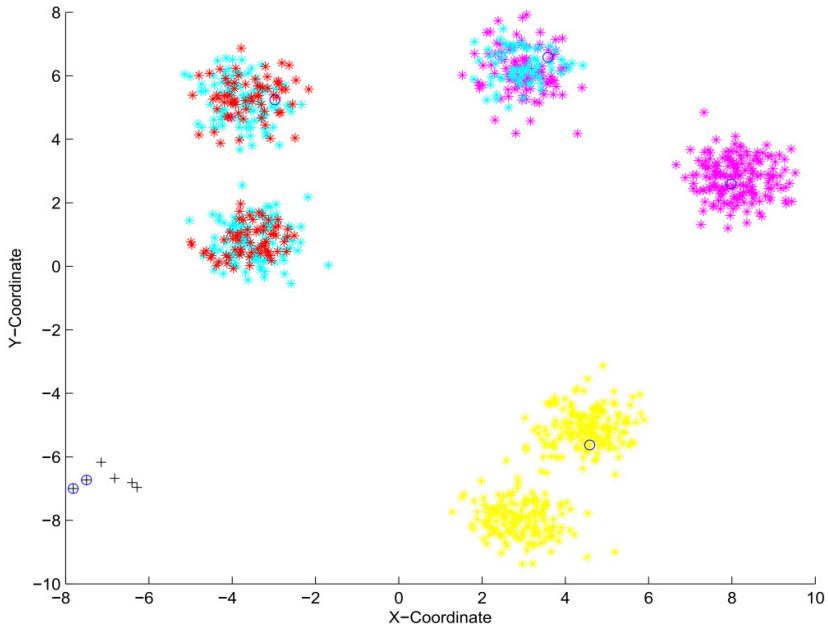

Figure 5. Classification using Online K−means forgetful (α=0.75) with initializing means randomly

We can see two different behaviors from the two learning rates. With a small learning rate, the algorithm acts very similar as the unforgetful one, and the clusters are severely overlapped. This is because that at the early stage, the prototypes move too slow thus, some points happened to get in their way would be clustered with them even if the points don't actually belong to them. A bigger learning rate seems to have a better performance in separating the clusters and spreading the prototypes, but the prototypes cannot finally converge. A big learning rate means a big influence by the newly-coming points, which would make the prototypes fluctuate a lot and lead to errors in the decision as we can see in the result.

Having noticed the difference in performances on the opening stage and the closing stages, we try to put the two methods together. The process can be separated into 2 periods, while in the first period a big learning rate is used to spread the prototypes as quickly as possible, and in the second period, a small learning rate is taken to converge the prototype. At first, prototypes would spread very quickly to avoid overlapping, and finally, a steady state would be reached.

In the following attempt, we use 300 points as the separation. In the first 300 points, the data are clustered under a learning rate of 0.75 and then change to 0.1.

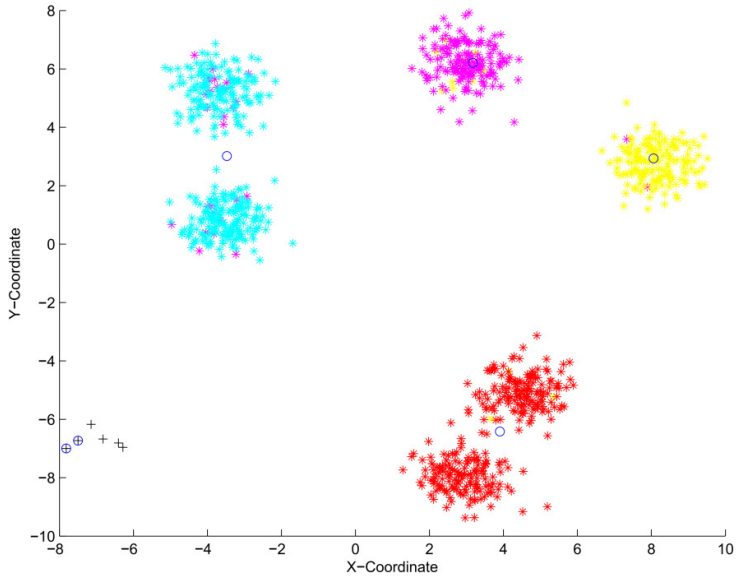

Figure 6. Classification using Online K−means forgetful (α_1=0.75;α_2=0.1) with initializing means randomly

Although the result is still not so convincing, but it's much better than the two forgetful ones at first sight. Actually, the unforgetful K-means is based on the same idea of decreasing the learning rate by steps, but this improved method stands out by its simplicity of calculation. We will use this modified model for the calculating and the analyzing during the rest of this paper.

3.1.2. Implementation of the BSAS Algorithm

We try the BSAS with only one threshold and see its performance. To prevent the number of clusters from exploding, we use an upper limit of 6 for cluster numbers. Set the threshold to 1, run the code.

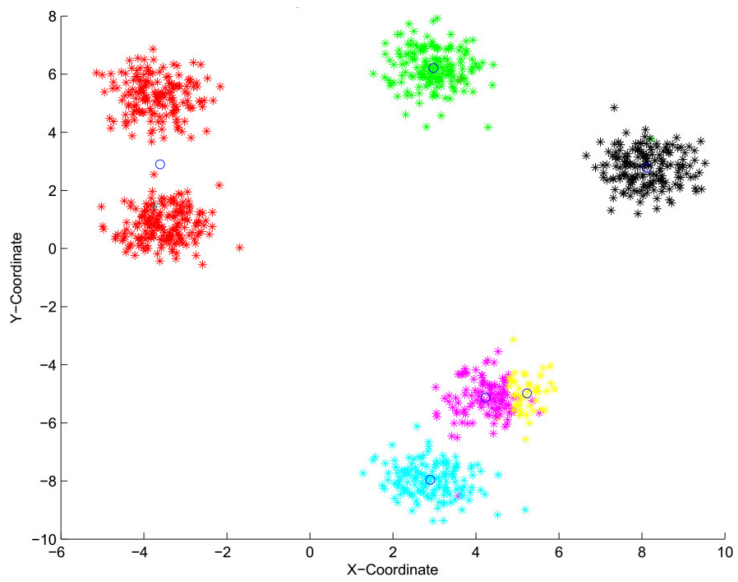

Figure 6. Classification using BSAS with θ=1 and maximum of 6 clusters

The performance looks bad. It seems that there are too many clusters as the threshold is too small. The threshold is increased to 3 and retest is done again.

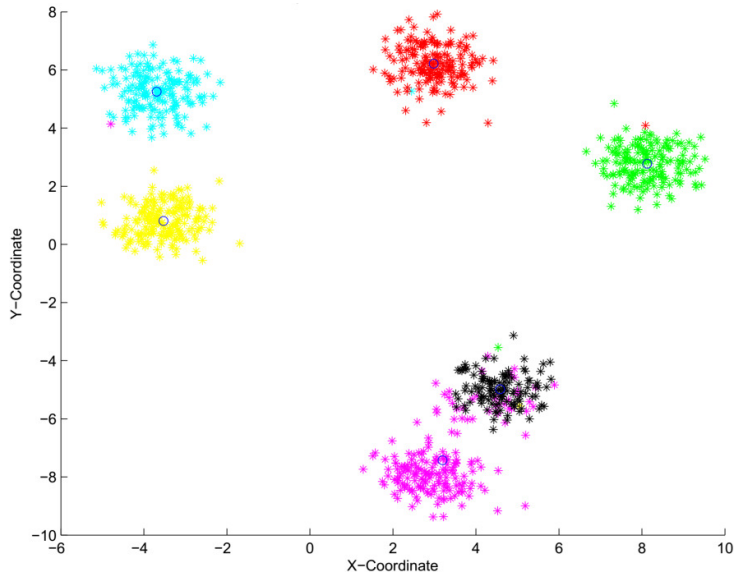

Figure 7. Classification using BSAS withθ=3 and maximum of 6 clusters

The performance is much better, but this needs the knowledge of the right value for the threshold apriori, which is not practical. In addition, in this test, we have not introduced the effect of the noise. We will see later in the paper the catastrophic results when noise is introduced, new misleading clusters will be created because of noise.

3.1.3. Implementation of the IWKO Algorithm

Let everything be the same except for the function to update the prototypes. In order to simplify the calculation, set n in the function to be 2.

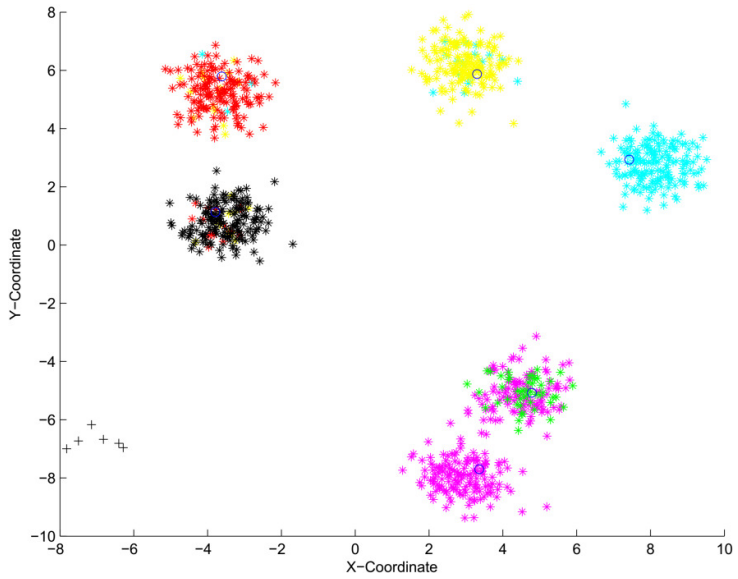

Figure 8. Classification using Inverse Weighted K−means with initializing means randomly

The algorithm managed to spread the prototypes and find all the clusters which the k-means could not. The drawback of this algorithm is that it needs several iterations at first as a learning phase where it can spread the prototypes. During this learning phase, the algorithm will make mistakes in allocating the coming data; this is evident by the mixed colors in one cluster. But once the spreading phase is done the algorithm starts making the right decisions.

3.1.4. Implementation of the KHMO Algorithm

The KHMO method is very similar to the former ones with the same idea of spreading all the prototypes in each step and with a simpler function.

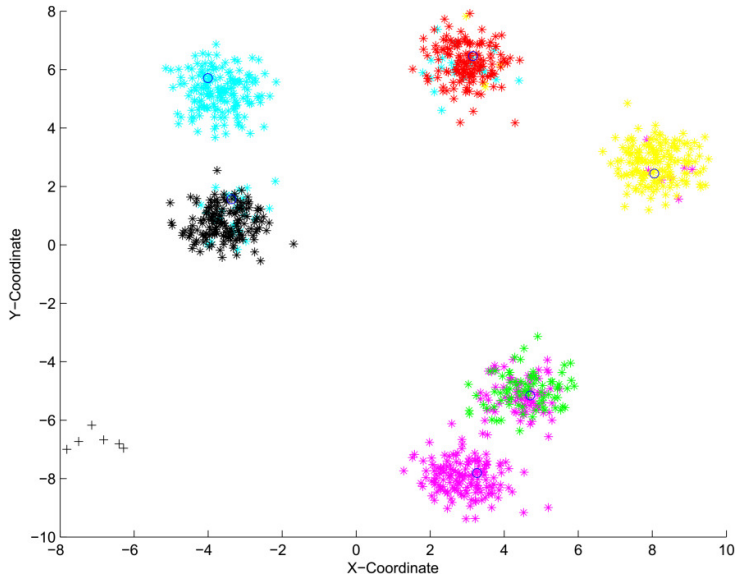

Figure 9. Classification using Online K−Harmonics means (learning coefficient=0.05) with initializing means randomly

We can see it does succeed at finding the clusters the same way IWKO found them but with less computation cost. The mixed points at the beginning also exists, showing another struggling beginning to decide on the clusters.

3.2. Study with Noise

3.2.1. Implementation of K-means Algorithm

Do the same execution of K-means to the data set with noise. For the forgetful one, we use the modified as it performs better without noise.

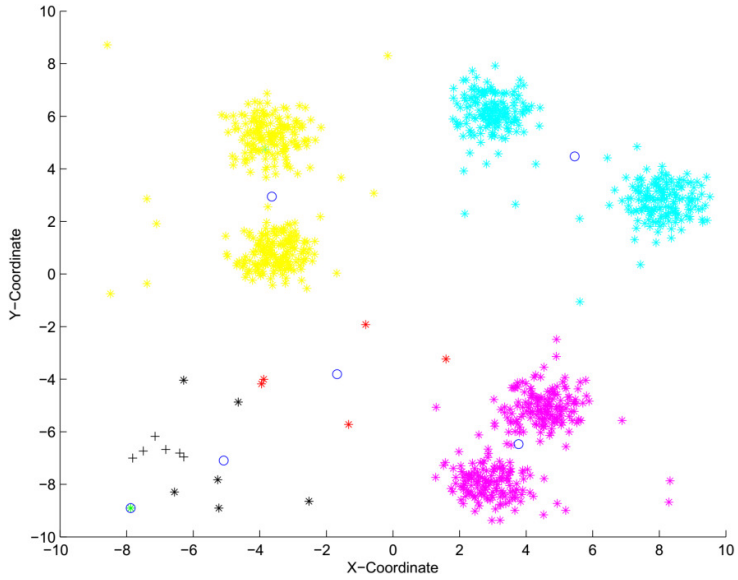

Figure 10. Classification using Online K−means unforgetful with initializing means Randomly

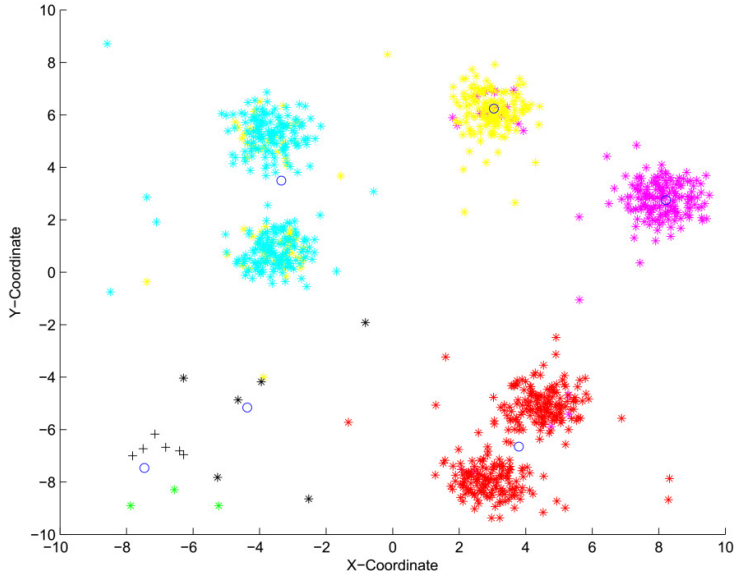

Figure 11. Classification using Online K−means forgetful (α_1=0.75;α_2=0.1) with initializing means randomly

It seems noise doesn't affect the performance of K-means a lot. When the amount of data is huge, a point of noise would have little effect on the mean. But no matter whether the noise exists, joint clusters appears. This is caused by the isolation between examples, with only one prototype - the one located nearest to the coming point - is changed. The truth is that if there are several clusters squeezed somewhere and the initial prototypes are selected far away, when one prototype is moved to this 'crowd', every point in the several clusters would be classified with this prototype as it is always the 'nearest'. This is why we need the IWKO algorithm.

3.2.2. Implementation of the BSAS Algorithm

According to the performance without noise, a threshold of 3 would be ideal in this case. Thus, we set the threshold to 3 to show the influence of noise better.

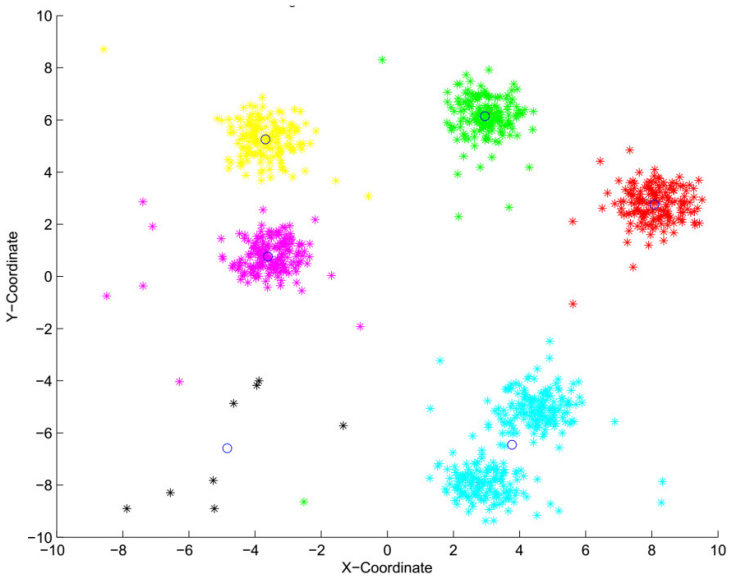

Figure 12. Classification using BSAS with θ=3 and maximum of 6 clusters

It turns out that the noise has a significant effect on this algorithm. The clusters are made quite perfectly without the noise, but in this case, two clusters at the bottom aren't even separated. It is reasonable by theory, as a newly-come noise would have a good chance to create a new cluster, which would disturb the performance.

3.2.3. Implementation of the IWKO Algorithm

The result is shown below.

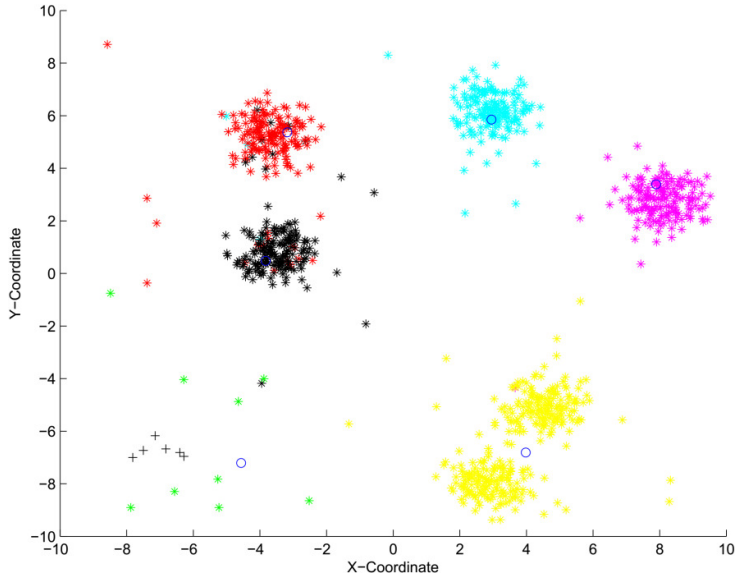

Figure 13. Classification using Inverse Weighted K−means with initializing means randomly

Surprisingly, we find that the IWKO is vastly influenced by the noise. Every point has the influence over all the prototypes, which means one single prototype has to 'tolerate' the harassment under all the noise. This would be quite a lot if comparing to the ordinary K-means.

3.2.4. Implementation of the IWKO Algorithm

The result is shown below.

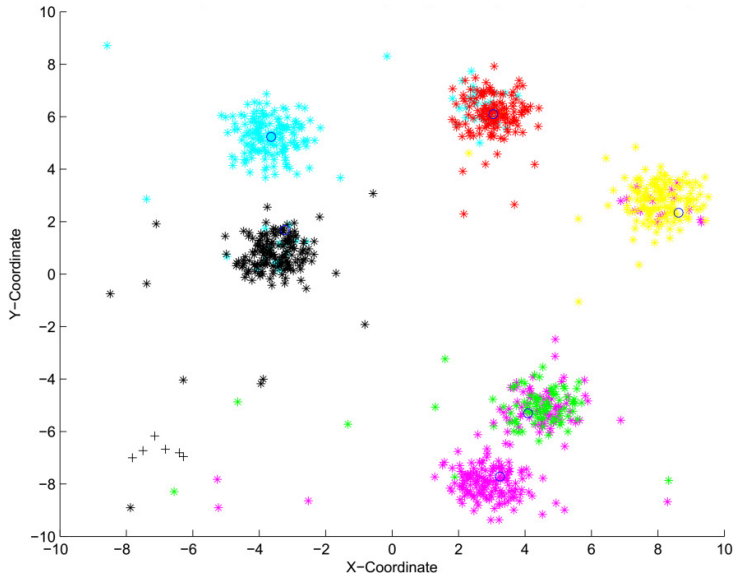

Figure 14. Classification using Online K−Harmonics means (learning coefficient=0.05) with initializing means randomly

Though similar to the IWKO on the idea of the algorithm, they have quite different robustness. The KHMO is not likely to be affected by noise, as the noise doesn't even change the result for clustering. The difference mechanism between KHMO and IWKO might have saved KHMO. The IWKO would find the nearest point and do the calculation, which means the calculation is different

from time to time. This gives the noises chance to influence the result with there location and sequences. While in the KHMO, every prototype changes under the same function. The noise would have a similar influence to all the prototypes wherever it appears, and the result is relatively honest.

4. ANALYSIS OF PERFORMANCES

The performance of algorithms is rated according to their execution speed, the accuracy of prototypes, the purity of clusters and the robustness.

- **The speed** is indicated by the time MATLAB needs to get the result. We run each algorithm several times, measure the time of each execution and take the average time of execution. Here, we have run each program 20 times and took the average execution time. As the speed of an algorithm is decided by the complexity of the algorithm itself rather than about the data coming inside, there is no need to measure the time with noise included.

- **The accuracy** is the distance between the real prototypes and the clustered ones. The indicator is calculated based on the average distance between each prototype after processing all the data and the true means that were generated at the moment of data generation. This is a one-to-one mapping between a prototype and its nearest true mean. The range of this indicator is a positive value, and the bigger the better with 0 a perfect value.

- **The purity** is an indicator of the percentage of points that are rightly classified. It is shown by the function[2]

$$\mathrm{P}(\omega, \mathrm{C}) = \frac{1}{N}\sum_{k} \max_{j} \left|\omega_k \cap C_j\right|$$

  where ω indicates the clusters (how actually the points are put together) and C indicates the classes (howexactly the points should be put together). N is the total number of documents that are correctly classified, and the value of sum should be the number of points that are correctly classified. To get a better illustration, we generate different data set each time and calculate the average purity as the purity of a certain algorithm. The indicator should be some value between 0 and 1, where 1 means perfectly clustered and 0 means the opposite. The noise is not considered inside.

- **The robustness** is the ability to keep off the influence of noise. An indication of robustness is based on the difference of velocity, accuracy, and purity between data sets with and without noise. The formation of cluster shown above can also help. The bigger the difference, the worse the robustness is. It can also be read from the graph in part 3.

Table 2. Performance of different algorithms.

Algorithms	Noise	Time	Accuracy	Purity	Robustness
K-means (forgetful)	0%	5.1119s	3.220	0.597	Good
	4%		5.012	0.601	
K-means (unforgetful)	0%	5.1593s	6.160	0.495	Normal
	4%		7.360	0.597	
BSAS	0%	5.2966s	0.207	0.875	Bad
	4%		1.540	0.739	
IWKO	0%	5.6401s	0.648	0.766	Normal
	4%		2.237	0.702	
KHMO	0%	5.2472s	0.612	0.789	Good
	4%		1.802	0.751	

We can have some comment on the result.

First, the velocity indicates how fast the system would respond to an input. It shows in the result that the forgetful K-means has the most rapid response and the IWKO has the slowest. It is consistent with what is shown by the theory. The IWKO and KHMO need to adjust position for every prototype on each step, so it's reasonable the two of them takes relatively more time, and the more steps of calculation made the IWKO even slower. The two K-means have to do nothing but calculating and adjusting only one mean at a time, thus they are relatively fast; the forgetful wins as it doesn't need to re-calculate the learning rate each time. At first, the BSAS seems confusing as its algorithm is quite simple without too many calculations. It turns out that it uses more 'if' judgments than the rest, which is quite costly.

Second, the accuracy shows how well-located the prototypes are. The results indicate that the normal K-means are working really bad on locating the prototypes. The best performance comes from the BSAS. This is because the BSAS has a good initial state by locating the point to neighborhoods of the real prototypes. It would be better to relocate the prototypes inside a small area rather than locate its step-by-step on the whole plane. The IWKO and the KHMO also stand out by their idea of adjusting every prototype rather than only one; thus, it's nothing strange to find them better than normal K-means.

Third, the purity is an indicator of how well the clusters are formed. The unforgetful K-means have the purity of less than 0.5, which means less than half of the data are clustered correctly. This is a disaster. The best performance comes from the BSAS because of the similar reason proposed in the part of accuracy, but its performance drops significantly after noises invade. The forgetful K-means seems almost not effect by the noise, but in general, the purity is too low. The IWKO and the KHMO have relatively more stable behavior, while the KHMO has a higher level of purity.

Last, the robustness of BSAS is really bad. The most robust algorithm is the forgetful K-means, with almost the same robustness. An interesting thing is that in the forgetful K-means algorithm, the purity is even better than that without the noise. It is probable that in this case, noise help to spread the initial prototypes, which would provide a better performance. We can also find out that the forgetful K-means doesn't change a lot with the noise.

5. CONCLUSIONS

In this paper, we studied 4 different algorithms of sequential clustering. We explained the theory, implemented them on self-generated data and analyzed their effectiveness. The implementation could prove some of the characteristics shown by theory, and direct comparison between the algorithms clearly reveals their pros and cons and preferred environment to be applied.

The sequential K-means stands out by its speed and robustness, as the mechanism and calculation behind the algorithm is really simple; but its accuracy is a disaster as a result. The BSAS let the system to start really near from the real prototypes, which would avoid a lot of error in the process of 'finding' the prototypes, making itself an ideal choice for a clean dataset; but it would collapse when noise is introduced. IWKO performs perfectly on getting accurate and robust results which is guaranteed by its meticulous calculation, but its obvious slowness prevents it from becoming the first choice. KHMO is quite moderate, doing well on very aspects with no prominent advantages or shortcomings, making it a good algorithm in general.

To sum up, we can conclude thatwithout the noise, the Basic Sequential Algorithmic Scheme (BSAS) would be the best algorithm among the four algorithms, but if noise is added, which is always the case in real-life systems, the K-Harmonic Means- Online Mode Algorithm (KHMO) would stand out with its robustness; if speed is the priority in a program, then the sequential K-means should be adopted, but if one still considers accuracy so important that speed can be sacrificed, IWKO would be a best choice.

Our research can be improved in its depth and width. In depth, the four algorithms have not been fully studied in this paper. The results are inevitably unstable as the generated data scale is too small while the algorithms are always implemented on big data circumstances; implementation on bigger datasets should make results more convincing. Two-dimension data are not common in real-life applications, and performance of the algorithms in higher-dimension datasets might be more accurate on results. In width, this paper only studied 4 algorithms but much more algorithms on sequential clustering could also be studied under a same methodology. Performance evaluation is still not strict, with several steps judged by sight, thus certain criteria should be constructed.

ACKNOWLEDGMENTS

The authors would like to thank our instructors and our universities.

REFERENCES

[1] S.Aghabozorgi ,A.Seyed Shirkhorshidi,and T.Ying Wah,(2015)"Time-series clustering - a decade review". Information Systems, vol.53, pp16-38.

[2] W.AshourBarbakh, Y.Wu, C.Fyfe, (2009)"Non-standard clustering criteria". In Non-Standard Parameter Adaptation for Exploratory Data Analysis, chapter 4, pages 49-72. Springer.

[3] T.Habib Sardar, Z.Ansari, (2018) "An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm".Future Computing and Informatics Journal. pp1-10.

[4] X.Huang, Y.Ye, L.Xiong, R.Y.K.Lau, N.Jiang, S.Wang,(2018) "Time series K-means: A new k-means type smooth subspace clustering for time series data". Information Sciences, vol.367-368, pp1-13.

[5] C.Yang, N.T.P.Quyen,(2018) "Data analysis framework of sequential clustering and classification using non-dominated sorting genetic algorithm". Vol.69, pp 704-718.

[6] X.Zhao, F.Cao, J.Liang, (2018) "A sequential ensemble clusterings generation algorithm for mixed data". Vol.335, pp 264-277.

[7] H. Manning, Prabhakar Raghavan. Evaluation of clustering. An Introduction to Information Retrieval. Cambridge University Press, 2008.

[8] Marcin Grzegorzek. Pattern Recognition Lecture: Sequential Clustering. Research Group for Pattern Recognition, Institute for Vision and Graphics, University of Siegen, Germany.

[9] Department of Computer Science Princeton University. Sequential k-means clustering. url: https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/C/sk means.htm.

AUTHORS

Xinchun Yang, undergraduate student from Department of Electrical Engineering, Tsinghua University, Beijing, China. Double major in economics in Peking University, Beijing, China. Exchanging at CentraleSupélec, Paris, France in 2018..

Wassim Kabbara, undergraduate student from Department of Energy Conversion, CentraleSupélec University, Paris, France. Double diploma with Lebanese University Faculty of Engineering, Electrical and Electronics Engineering.