Stabilization of Fully Dynamic *k*-Center Clustering

Project Plan

3035123655 Choi Jae Won
3035123722 Ha Tae Min

28/09/2019

1 Clustering algorithms and applications

Clustering is the task of grouping a set of objects in a way that objects in the same group (cluster) are more similar to each other than to those in other groups. There are many different ways to decide whether an object belongs to one group or another, but most of the clustering algorithms use quantitative or qualitative measures to draw lines between one group and another. Given attributes of input objects, clustering algorithms try to group them accordingly, which ultimately results in labeled clusters with similar properties within the group as in **Figure 1**.

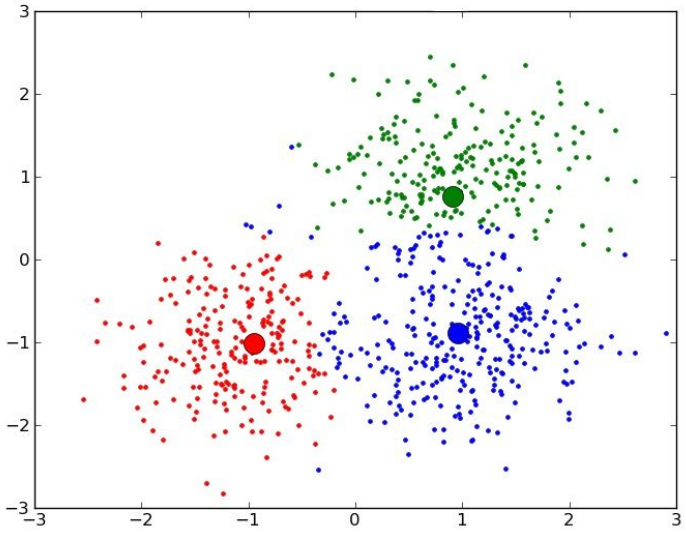

Figure 1. Example of 3-Center Clustering [3]

Data points are scattered in cartesian coordinates and thus scattered points can be clustered around selected centers.

Today, over 6000 tweets are posted every second on Twitter, 60000 queries are handled every second by Google, and more than 400 hours of Youtube videos are uploaded every minute. With such a huge volume of data flowing in and out of the internet every second, there are numbers of demands for clustering algorithms that can group and cluster various data responsively and accurately. In this manner, Fully Dynamic k-Center Clustering provides a dynamic algorithm with arbitrary insertion and deletion that can update the clusters of data objects dynamically.

1.1 Fully Dynamic *k*-Center Clustering

In simplistic clustering models, datasets are static without any future changes implied. In this setting, there is no need for any dynamic models to support insertion and deletions of data points. However, in many real-world problems, one might deal with insertions and deletions that can affect the whole structure of clusters if a substantial part of data is added or removed. In *Fully Dynamic k-Center Clustering* [1], the algorithm starts with selecting a random point to be a candidate for one of the *k*-centers. For each random center candidate, β (proposed optimal radius of a cluster) is approximated using (2+ε)-approximation. Then, from the unclustered data points, the new center is selected at

random and the following step repeats. If k clusters are formed successfully by using these parameters, the algorithm terminates, but if it fails to include all data points, the algorithm selects a new random center, and test for a newly approximated β radius.

1.2 Arbitrary Insertion and Deletion

For the insertion, the data point that is to be inserted to the clusters is compared to the pre-existing clusters. If the centers' radius can include the new data point's position, it is inserted to the particular cluster. However, if all the clusters cannot bound the new data point, the algorithm creates a new center and include the data point to the cluster. If there are k clusters predefined such that a new center cannot be created, the data point is classified as unclustered data point.

For the arbitrary deletion, the simple case is when the deleted point is not one of the k centers, where the deletion can remove one data point from the cluster in $O(1)$ time complexity. However, when the deleted point is one of the k centers, some data points lose their center and thereby should be re-clustered afterwards. For clusters that are formulated after the deleted center's cluster, should all be re-clustered by selecting uniformly random center points. This operation makes the amortized time complexity of the algorithm to be bounded by $O(\frac{k^2 log(\delta)}{\varepsilon})$.

2 Objective

The following chapter introduces the overall objectives of the project. Stability of the cluster is defined to clarify the important aspect of the existing algorithm, which includes the areas of improvements. Furthermore, in regards to the stability, more specific problems and solutions are introduced and subsequently narrowed down to main feasible objectives of the project.

2.1 Definition of Stability

Due to the nature of the fully dynamic k-center clustering that inserts and deletes the data points dynamically, the clusters are prone to be updated by these operations more often and refactor their structures. Therefore, clusters obtained by dynamic algorithms have a higher chance of redefining the "key" points (centers of the clusters) than those generated by static algorithms. Reassignment of these key points lead to reconstructions of the clusters, and thereby new clusters would contain different data points.

Consequently, the new clusters are, for being formulated by different members and thereby losing previous information, unstable. Hence, stability is how similar the new clusters are to the previous clusters (clusters before re-clustering), where similarity of the clusters can be measured with different metrics. Numerous measures for comparing clusterings have been proposed. Besides the class of *pair-counting based* and *set-matching based* measures, *information theoretic* measures form another fundamental class (Nguyen Xuan Vinh, Julien Epps and James Bailey, 2010). With these measures, the similarity of new clustering and old clustering can be assessed, thereby defining stability of clustering.

2.2 Problem and Solution

The main reason that the fully dynamic *k*-center clustering results in unstable clusters is that the algorithm performs re-clustering if any of the key points is deleted (See **Figure 2**).

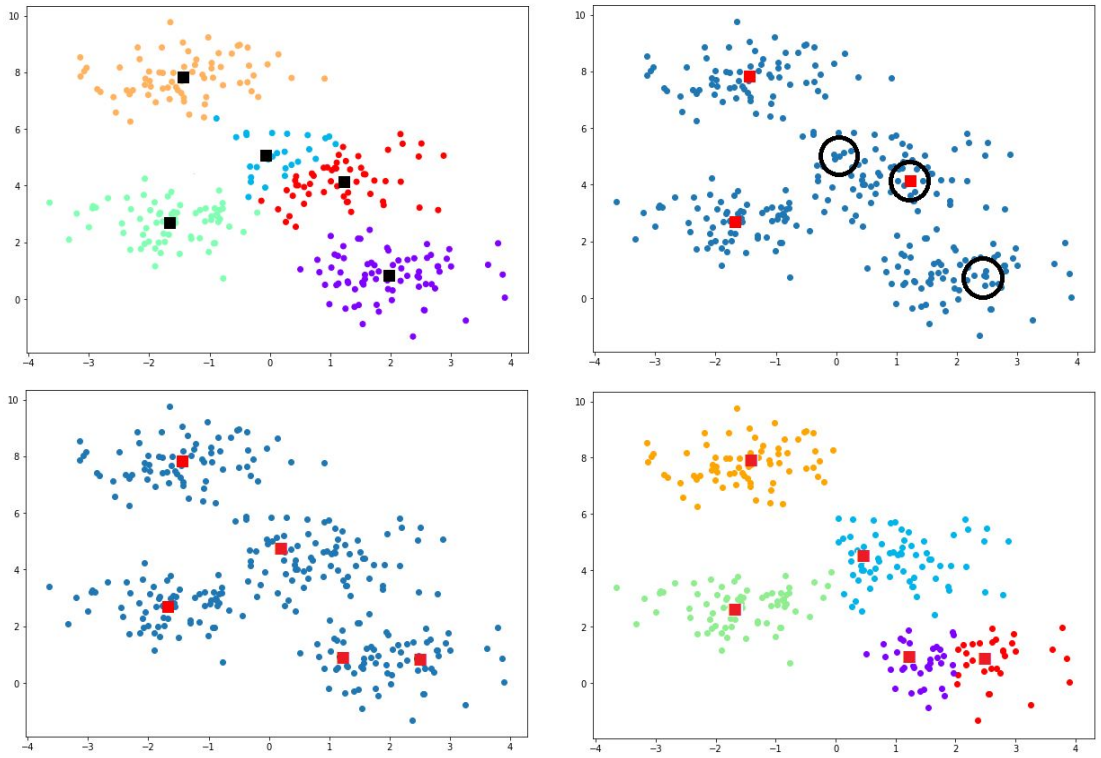

Figure 2. Sample Run of Fully Dynamic *k*-Center Clustering Deletion and Re-Clustering

When re-clustering is performed, new centers of the clusters are selected uniformly at random from unclustered data points. That is, the new key points are selected without any correlation to the previous key points. This randomness would induce the arbitrary formulation of the clustering which shares no similarity with the formulation of previous clustering, thereby resulting in unstable clusters.

To ensure stability, the random selection of centers can be replaced with a prefered selection of centers when re-clustering is performed. That is, the stability can be ensured if similar clusters can be formed whenever clusters have to be re-clustered (See **Figure 3**).

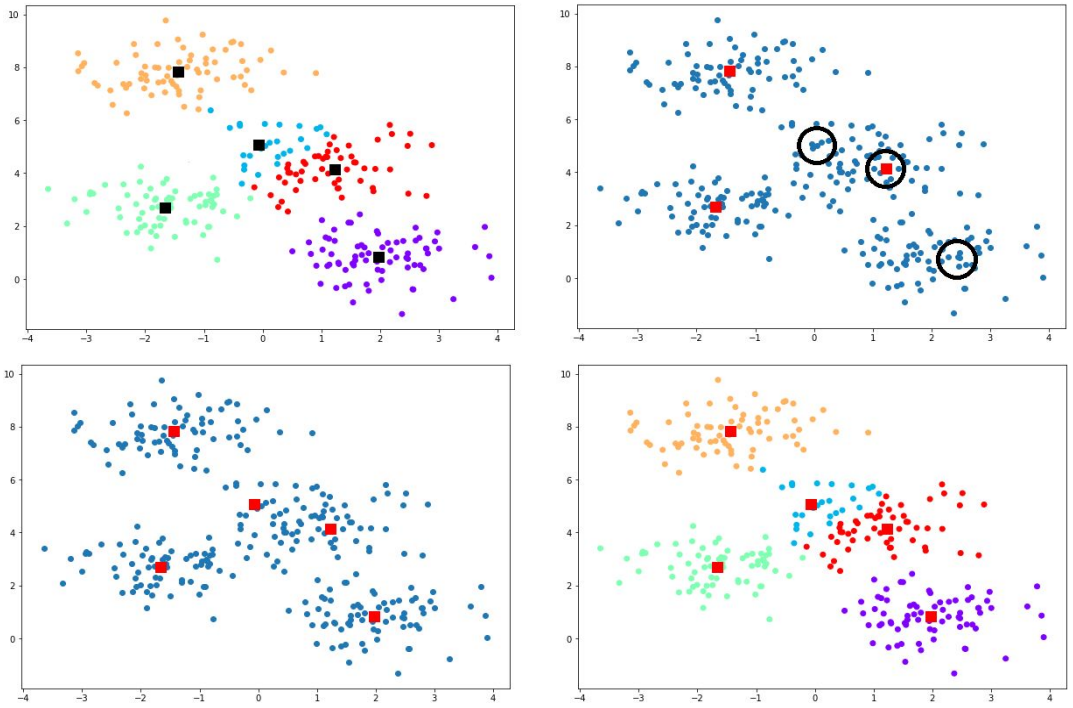

Figure 3. Sample Diagram for the Stability-Improved Algorithm

As in **Figure 3**, stability-improved algorithm will choose the similar centers with the original cluster before deletion. The method to select preferred center will be outlined in the methodology chapter.

2.3 Main Objectives

Considering the current status and limitations of the recent research papers, main objectives of the project can be narrowed down as follows:

(1) Implement the existing algorithm based on *Fully Dynamic k-Center Algorithm* to a more extendable form in other programming language (Python) so that enhancements can be easily built upon.
(2) Understand different existing metrics from recent research papers for measuring "stability" of the clusters and implement the metric algorithms based on them.
(3) Implement the stable algorithms (stabilization of the clusters after re-clustering by finding the good candidates for the new centers of the clusters).
(4) Compare the results, using stability measuring metrics, of the existing algorithm and advanced algorithms. Based on the results of different metrics, improve the stability of algorithms.
(5) Visualize the advanced algorithms and existing algorithms using frameworks (tensorboard or others) for multi-dimensional datasets.

3 Methodology

The project will mainly focus on experimenting proposed theories that can stabilize the existing algorithm. To achieve this goal, programming the existing paper's algorithm (Fully Dynamic *k*-Center Clustering) is essential. After implementing the existing algorithm, we will define how to measure the stability of the *k*-centered clusters. This step will require theoretical justifications for comparing similarities of each cluster. Finally, after the definition of stability and the methods to measure stability are well established, the changes in the new algorithm can then be tested and measured. The testing environment will be built upon the existing codes introduced in *Fully Dynamic k-Center Algorithm*.

3.1 Software and Hardware Descriptions

Algorithms used in this paper are implemented in Python due to its flexibility. Python will provide libraries that can easily link the existing C language program for Fully Dynamic *k*-Center Clustering to our system. All of our experiments will run on a personal computer with Intel(R) Core i7-8750H with 2.20GHz, GeForce RTX 2060 with 16GB RAM.

Publicly available datasets will be considered as well as the data collected from Twitter as shown in **Table 1**.

Name	Source	Type	#Updates
Twitter	twitter.com	2D points	42M
Flickr	yfcc100m.appspot.com	2D points	96M
Porto taxi trajectories	archive.ics.uci.edu	Trajectories	83M

Table 1. Datasets statistics

3.2 Implementation of *Fully Dynamic k-Center Clustering*

Although Python is used as a medium language for the project, most of the core functions are already available in C language. Components that can be imported via wrapper libraries, such as Cython or Pyobject, will be integrated into our system. To measure the successful integration of the components, the portions of the datasets used in *Fully Dynamic k-Center Clustering* (Twitter, Flickr, and Trajectories) would be used to simulate the experiment conducted in the paper. As in the paper, approximation ratio as a function of k and ε would be calculated and results of fully dynamic algorithm and static algorithm would be compared. These results would be compared to check if the same output could be generated in different system. Integration would be considered to be successful if the values of approximation ratio and comparison of static and dynamic algorithm calculated from our system are identical to those from the paper.

3.3 Definition and Measurement of Stability

The definition of stability is not defined for the *k*-center clusters. Recent research paper (*Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance*) mainly discusses about methods to measure similarities between different clusters. *Pair-counting*, *set-matching*, and *information theoretic* measures are the classes that will be tested on our system. As mentioned in the paper, the *normalized information distance* (NID) would fit our purpose of comparing different clusters. NID is calculated as follows (see **Table 2**).

$\mathbf{U}\backslash\mathbf{V}$	V_1	V_2	...	V_C	Sums
U_1	n_{11}	n_{12}	...	n_{1C}	a_1
U_2	n_{21}	n_{22}	...	n_{2C}	a_2
⋮	⋮	⋮	⋱	⋮	⋮
U_R	n_{R1}	n_{R2}	...	n_{RC}	a_R
Sums	b_1	b_2	...	b_C	$\sum_{ij} n_{ij} = N$

Table 2. The Contingency Table [2], $n_{ij} = |U_i \cap V_j|$

U and **V** denote the random clusterings generated by the algorithm. Each element in the table, n_{ij} denotes the element that is in both **U** and **V.** Furthermore, given **U** and **V,** their entropies, joint entropy, conditional entropies and mutual information (MI) are defined respectively as follows:

$$
\begin{aligned}
H(\mathbf{U}) &= -\sum_{i=1}^{R} \frac{a_i}{N} \log \frac{a_i}{N}, \\
H(\mathbf{U},\mathbf{V}) &= -\sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}, \\
H(\mathbf{U}|\mathbf{V}) &= -\sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{b_j/N}, \\
I(\mathbf{U},\mathbf{V}) &= \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{a_i b_j/N^2}.
\end{aligned}
$$

Suppose we need to transmit all the cluster labels in U on a communication channel, then $H(\mathbf{U})$ can be interpreted as the average amount of information, for example, in bits, needed to encode the cluster label of each data point according to **U**. Now suppose that **V** is made available to the receiver, then $H(\mathbf{U}|\mathbf{V})$ denotes the average number of bits needed to transmit each label in **U** if **V** is already known. We are interested in how seeing how much $H(\mathbf{U}|\mathbf{V})$ is smaller than $H(\mathbf{U})$, that is, how much the knowledge of **V** helps us to reduce the number of bits needed to encode **U**. This can be quantified in terms of the mutual information $H(\mathbf{U}) - H(\mathbf{U}|\mathbf{V}) = I(\mathbf{U},\mathbf{V})$ (Nguyen Xuan Vinh, Julien Epps and James Bailey, 2010). Then, these information lead to the metrics which define the stability of the clustering of the project (see **Table 3**).

Name	Expression	Range	Metric	Related sources
Normalized distance measures				
d_{joint} (Normalized VI)	$1-\frac{I(\mathbf{U},\mathbf{V})}{H(\mathbf{U},\mathbf{V})}$	[0,1]	✓	Kraskov et al. (2005)
d_{max} (Normalized Information Distance)	$1-\frac{I(\mathbf{U},\mathbf{V})}{\max\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	✓	Kraskov et al. (2005)
d_{sum}	$1-\frac{2I(\mathbf{U},\mathbf{V})}{H(\mathbf{U})+H(\mathbf{V})}$	[0,1]	✗	
d_{sqrt}	$1-\frac{I(\mathbf{U},\mathbf{V})}{\sqrt{\{H(\mathbf{U}),H(\mathbf{V})\}}}$	[0,1]	✗	
d_{min}	$1-\frac{I(\mathbf{U},\mathbf{V})}{\min\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	✗	

Table 3. Information Theoretic-Based Distance Measures [2]

As in **Table 3**, *normalized variation of information,* d_{joint}, and *normalized information distance,* d_{max} are proved to be adequate metrics to measure the information sharing (similarities) between two different clusterings, thereby becoming proper methods to define the stability of the clusters.

3.4 Implementation of Stable Algorithm

To minimize the randomness while maximizing the chance of formulating the "similar" clusters, new centers of the clusters must be selected based on the previous centers of the clusters (centers of clusters before re-clustering). One of the possible methods of picking the new candidates for key points is to pick the data points that are closest in distance to the previous key points. A key point and the closest data point are in the same cluster by high chance. If the same value of the radius of the cluster is taken and the closest data point is taken as a new center of the cluster, a newly generated cluster is likely to contain the data points that were in the same cluster as the new center of the cluster, especially when the data points are densely distributed. This ensures the stability measured with *information theoretic* similarity and distance measures by having possibly the most similar data points.

The criteria for selecting the good candidates for the new key points could be further studied and implemented accordingly after further discussion of the methods for measuring the clustering similarity and assessment of the feasibility.

3.5 Testing of Enhanced Algorithm

The test will rely on the metrics mentioned in 3.3. The project aims to reduce the arbitrary outcomes after the re-clustering. Therefore, the test cases will be measured against the change in stability as defined in 3.3. If the new algorithm can provide better metric scores of measured stability and minimize the randomness, it would be considered as a more stable algorithm.

4 Project Schedule and Milestones

It is now assumed that the main idea of *Fully Dynamic k-Center Clustering* is understood and the scope of the limitations is defined. Accordingly, it is essential to subdivide the important milestones in time to keep track of the progress of the overall project. The project can be separated into four main phases (See **Table 4**).

Phase	Date	Objectives
Phase 1 Implementation	**Sep - Oct**	● Thoroughly understand the *Fully Dynamic k-Center Clustering* and implement the codes in Python to have the same result as the paper.
Phase 2 Stability Metrics	**Oct - Dec**	● Fully understand the *Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance* and implement the codes in Python. ● Test with algorithm implemented from phase 1 with different metrics, especially *pair counting based, set-matching based, and information theoretic* to test by which measures the existing algorithm is unstable.
Phase 3 Enhancement	**Dec - Feb**	● Devise and implement the different algorithms to stabilize the clusters of existing algorithm. ● Test whether the introduced algorithm can enhance the stability with different metrics. ● Based on the results, revert to phase 2 to come up with better algorithms.
Phase 4 Results	**Feb - Apr**	● With introduced algorithms and metrics, summarize which algorithms outperform the others when measured with various metrics. ● Mathematically prove the reason that certain algorithms outperform the others with different metrics.

Table 4. Project Schedule

For Phase 1, the sample codes are now available in C. Python will be used as a main programming language for extensibility and further enhancements. For phase 2, due to the limitation of the existing codes, a new program for stability metrics will be built from scratch by referring to several sources. To ensure the correctness, the datasets used in the paper (Microarray datasets from Monti et al. (2003)) will be used to test whether the implemented codes output the same results. For phase 3, the most plausible approach would be to pick the closest data point as the new center of the cluster before re-clustering. However, after the actual implementation of the algorithm and measurements of the results, it is likely to revert to phase 2 to come up with a better algorithm. Presumably with the reasonable and satisfiable results, all the steps and outcomes will be summarized in phase 4.

5 References

[1] T-H. Hubert Chan, Arnaud Guerqin, and Mauro Sozio. *Fully dynamic k-center clustering*. In Proceedings of the 2018 World Wide Web Conference, WWW ’18, pages 579–587, 2018.

[2] Nguyen Xuan Vinh, Julien Epps, James Bailey. *Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance.* Journal of Machine Learning Research 11, pages 2837-2854, 2010.

[3] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," in *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32-40, January 1975.