Comparing Trajectory Clustering Methodologies

Selcuk Gulcan
Computer Science Department of
Bilkent University
Ankara, Turkey
selcuk.gulcan@bilkent.edu.tr

***Abstract*—Recent rise in geographic trajectory data increased need for methodologies to analyze such data. Clustering is common practice in the case where no labels are available. Clustering trajectories is not straightforward due to their high dimensionality and their varying length. Hierarchical, distribution based, partition based and density based approaches has been offered in the literature. One specific algorithm from each group has been applied on civil flight dataset in order to compare these approaches. Implementation approaches of k-medoids, DBSCAN, gaussian mixture model and HDBSCAN algorithms and comparison of these methods are discussed in the paper.**

***Index Terms*—trajectory, clustering, dbscan, k-medoids, gaussian, mixture, hdbcan.**

I. Introduction

Capturing motion characteristics of moving objects is an interest of many areas including navigation, surveillance, tracking and anomaly detection[1]. Object movements are generally shown with trajectories. Trajectory can be defined as a list of position, time pairs in a N dimensional space [2]. Depending on dataset, trajectories may have varying length. This property of trajectories make them difficult to analyze with classic clustering methods. Clustering approaches can be divided into four category: Hierarchical, distribution based, density based and partition based.

The goal of the project to apply k-medoids, DBSCAN, gaussian mixture model (GMM) and HDBSCAN algorithms on civil flight trajectory dataset. Each of these algorithms correspond to a trajectory clustering approach. k-medoids algorithm partitions dataset into different regions and cluster dataset by assigning trajectories into them. This is one of the partition based approach. DBSCAN algorithm find places with high density and tries to expand this place to form a cluster. Gaussian mixture model assumes there is some underlying distribution on feature space. HDBSCAN algorithm on the other hand first tries to cluster data into large cluster then divides these cluster into subclusters.

The paper is organized as follows: Section 2 covers related studies in the literature. Dataset is described in section 3. Precomputation steps which include data representation and pairwise distance function, are mentioned in section 4. Discussion of selected algorithms is given in the following sections. Finally section 6 covers comparison and study results.

II. Related Literature

Trajectory modeling and pattern recognition studies on trajectory data are trendy topics in literature. Various methods are reported in the papers. Calculating similarity of different trajectories is important part of most methods. Since trajectory is not simply a point in space but list of points, classic distance functions such as euclidean and haversine distance cannot be applied directly. Some specialized functions such as dynamic time warping[3], symmetric euclidean distance[4], longest common subsequence[5], hausdorff distance[1] are proposed to calculate similarity between pair of trajectories. Different trajectory representations are also documented. This paper [6] uses flow vector representation where each tuple in trajectory contain x, y coordinates and relative direction vector for x and y axis to find common hurricane trajectory samples. Another approach[7] used a representation using only direction and speed information of aircraft trajectories and applied kernel density estimation. Although methods are presented for different set of domain such as hand movement detection, hurricane trajectory clustering, pedestrian movement offered solution can be applied any kind of trajectory data regardless of where they are obtained from.

III. Dataset Description

Flight data is scraped from real time flight information provider websites. Data is consists of a set of input files labeled with timestamp information. Each file has flight information of planes which are detected at that timestamp. This flight information contain coordinate, ground speed, altitude, vertical speed, departure and arrival airports and identification number of plane but we just use spatial information.

Total trajectory count: 31546
Average trajectory length: 440
Trajectory frequency: 8 sec
Dataset corresponds around 2 week flight data on Turkey.

IV. Precomputation

Since each file is a snapshot of flights at that moment (Figure 1), points of a trajectory is spread on multiple files. We gather these points together so that points of a trajectory is included in a single file. All distance units is converted to meter and speed units are converted to meter per second to make features consistent with each other. We observed that some trajectories just contain 0-20 points, they are considered as noise and removed. The other observation is that planes stay

Fig. 1. Each file contains a snapshot of air traffic

in the airport for a long time before they take off. These parts are also removed to achieve better running time performances.

A. Hausdorff Distance

Bidirectional hausdorff distance of set A to set B is defined as maximum of minimum distance for each element in A to each element in B [4]. Since, this definition is not symmetric hausdorff distance calculates distance from B to A as well and then return maximum of these.

$$H(A,B) = max(d(A,B), d(B,A))$$

$$where$$

$$d(A,B) = max_{a \in A} min_{b \in B} ||a - b||$$

This definition uses an inner distance function calculate distance between two points. Our dataset contain geographic coordinates so haversine distance should be a good choice. However, our toolset does not support that and mercator projection gives good approximation between closer points so we decide to use euclidean distance.

Most clustering algorithms calculate similarity between each pair at least once. That means $O(n^2)$ time complexity. We decide to calculate distance matrix beforehand and save to be able use it quickly whenever it is needed. In this case $O(n^2)$ storage is needed. To overcome this problem, we decided to use sparse matrices. Most trajectory pairs are very far from each other so we can consider their distance as infinity instead of their true distance. Then we only store distance which is less than a certain threshold (1000 km). We tweaked clustering algorithm to handle such sparse distance matrices.

B. Trajectory Segmentation

Trajectory contains a point for each 8 seconds of flight. It increases number of data point a trajectory have and because of it, distance calculations take a while. Trajectory lengths should be reduced to overcome this problem. Instead of resampling trajectories with a lower frequency, We decided to remove uninformative points. Points relying on a straight line do not say much about pattern of the trajectory, so points having the same direction information with previous point are removed. First and last point of trajectories are always preserved. This process is reported on [3] to identify sub trajectories but here our sole purpose is to calculate distances faster. Figure 1 shows an example of trajectory segmentation applied on a synthetic trajectory. Although most points are removed, the shape of trajectory is still conserved.

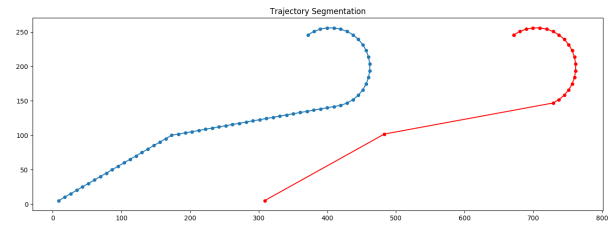

Fig. 2. Trajectory segmentation on an artificial trajectory

After trajectory segmentation, average number of point in trajectory is reduced from 440 to 19.5. However, having the same distance results is also important. We took an set of 1000 trajectory to test this. Only 1 percent of distance values are changed. We accepted this tradeoff to speed up clustering process. 1000 sample from resulting trajectories are depicted in Figure 2.

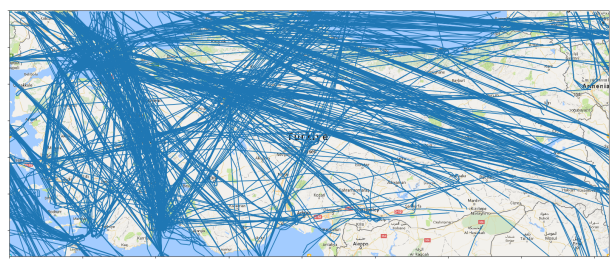

Fig. 3. Trajectories depicted after preprocess

V. Clustering Algorithms

Five clustering methods are considered and four of them are implemented. Here are the considered algorithms:

- K-Means: We should be able to calculate mean of set of data to perform this method. For points in euclidean space, mean of points is well defined but it is hard to define and calculate for trajectories. Instead of inventing a definition, we look for other methods.
- K-Medoids: Very similar method with K-Means, but we pick centroids instead of calculating means.
- Gaussian Mixture Model: We do not calculate pairwise distances unlike other methods. K key points are selected from each sample to present whole trajectory. These points are assumed to be features of the dataset and coming from an unknown gaussian distribution functions. So the whole dataset is assumed to be generated from a certain number of distribution. Clusters are formed according to likelihood of samples generated from a gaussian distribution.

- DBSCAN: DBSCAN saves us from to estimate number of cluster. DBSCAN clusters data based on minimum sample in each group and similarity threshold, epsilon. DBSCAN can also detect noise data. Noise data is trajectories which are not assigned to any cluster.
- HDBSCAN: HDBSCAN answers some limitation of DBSCAN algorithm. It clusters areas with high density as DBSCAN does. However, it also forms a hierarchy while doing so. Main motivation of applying this method is to be able to capture cluster with varying density.

A. K-medoids

It works very similar to K means algorithm. They both try to minimize the distance between trajectories in a cluster and center trajectory. They form K clusters by doing so. In K-means algorithm this center trajectory does not have to be in the dataset. It is calculated as computing averages of samples in the cluster. In K-medoids method however, a sample in the cluster is chosen as medoids (or centroid). Distances which is attempted to be minimized are calculated according to these centroids. As in K-means algorithm, centroids are reassigned at each iteration until a convergence is reached or number of iteration exceeds a threshold.

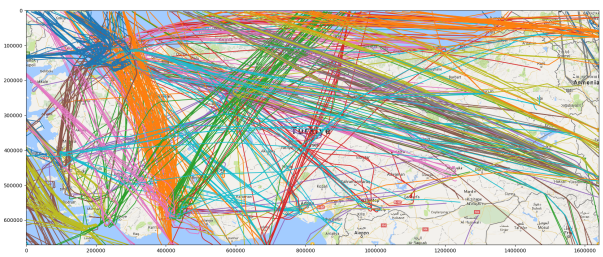

Fig. 4. k-medoids when $K = 30$

Main question is how can we choose number of clusters (K). Large K value may separate similar trajectories into different cluster. This situation can be seen in Figure 4. Although some trajectories are very close to each other, they are colored differently. On the other hand, partitioning with small K may not capture some clusters and different cluster may be merged as it can be seen in Figure 5.

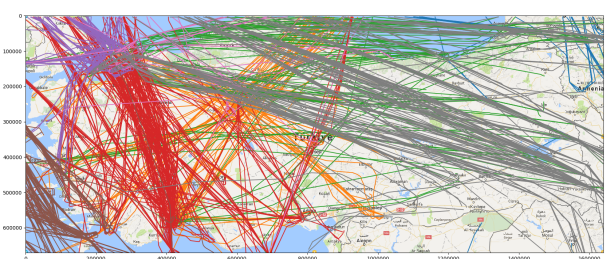

Fig. 5. k-medoids when $K = 8$

We might choose this parameter by visually inspecting dataset. This approach could be reasonable for a small dataset. However our dataset is large and trajectory projections become chaotic since each sample is not a single point but list of many line segments. When we look at Figure 1, it can be seen that it is difficult to guess K parameter by just looking at unlabeled trajectory plot.

Another approach is to apply elbow method for sum of squared error (SSE) with various K values. SSE value for a particular clustering with centroids $(c_1, c_2, ...c_k)$ is defined as

$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(x, c_i)^2$$

We plot SSE values with $1 < K < 80$ values in Figure 6. It is expected that SSE values decreases as K becomes larger. However this change is not dramatic when $K > 20$. So set number of cluster (K) parameter to 20.

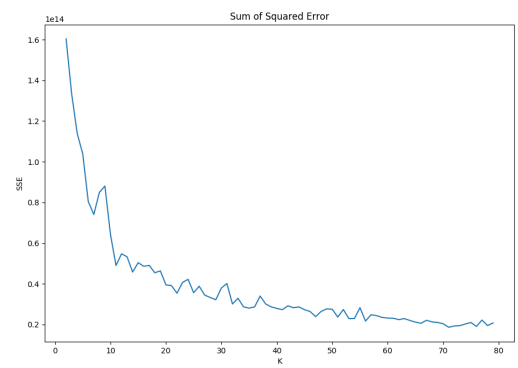

Fig. 6. Elbow plot to choose parameter K

B. Gaussian Mixture Model

We have to change our trajectory representation to use GMM. Here we introduce k-key point trajectory representation.

$$traj =< (p_1, t_1), (p_2, t_2), ..., (p_k, t_k) >$$

where p_i is point of plane at time t_i. First point is assigned to p_1 and last point is assigned to p_k always. Another constraint is that difference between each consecutive time values must be same for a trajectory.

$$t_k - t_{k-1} = t_{k-1} - t_{k-2} = ... = t_3 - t_2 = t_2 - t_1$$

Then vector $< p_1, p_2, ..., p_k >$ is used as feature vector in gaussian mixture model. We expect points at the same index to form a gaussian distribution for similar trajectories. That is the underlying motivation for this representation.

Two parameters are directly affecting resulting clustering. First one is the number of cluster. Akaike information criterion (AIK) and bayesian information criterion (BIC) are used to tune this parameter. Another parameter is the length of feature vector (k). Small k value may fail to capture pattern of trajectory and large k values may cause some trajectories to be

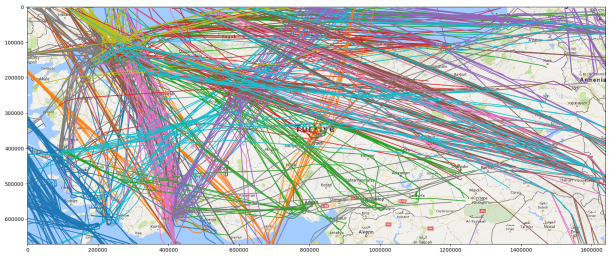

Fig. 7. Clustering with GMM

assigned to a distant cluster, especially when number of cluster is small. So k value must be tuned as well. By trajectory definition, length must be greater than 2. Also dataset may contain trajectories whose length is 20. Therefore k value must be between 2 and 20 inclusive. We do not set a single value for this parameter. We try 2, 5, 10 and 20 and report the results.

C. DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) finds a set of points which have high density. Then, expands this set by adding similar points [8]. Unlike previous methods, DBSCAN does not expect number of clusters to be given. Instead, min samples and ε parameters are provided to DBSCAN. Points are assigned as core, border or noise points. Linked (ε distance reachable) core and border points form a cluster while noise points are labeled as outlier and are not included in any of clusters.

There is an example in Figure 8 to show how DBSCAN results on a synthetic trajectory set. Four group of trajectories are generated. Each group consists of 10 similar trajectories. Then 5 random trajectories are added to the set in order to represent outliers. Red trajectories represent outliers in the figure.

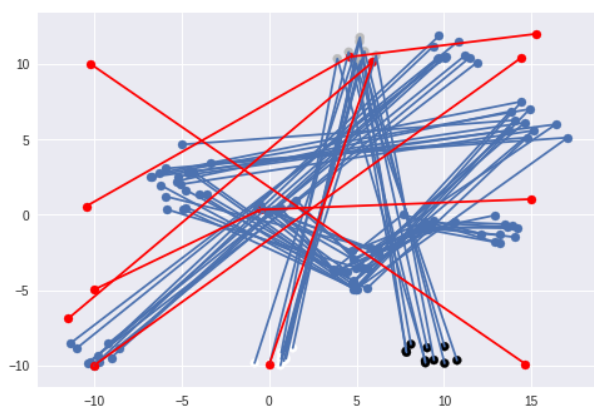

Fig. 8. 45 synthetic trajectories with 5 outliers

After tuning parameters, DBSCAN successfully identify outlier trajectories (drawn as black in Figure 9).

Then, method is applied on real dataset. Epsilon (120 km) and min samples (50) parameters are chosen using visual inspection. 44 clusters are formed. 7 percent of trajectories are

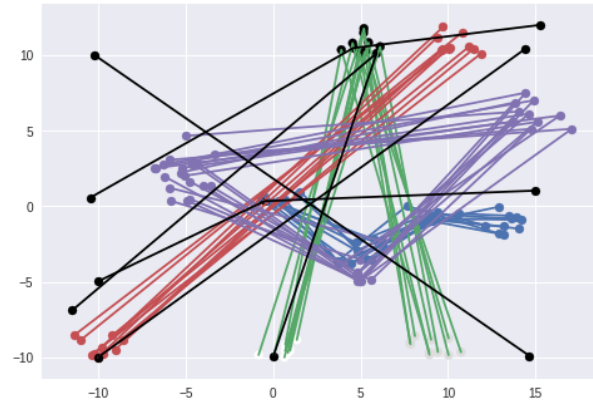

Fig. 9. Clusters of trajectories and black outliers

identified as noise. Figure 10 shows resulting clusters, noise trajectories are removed from figure.

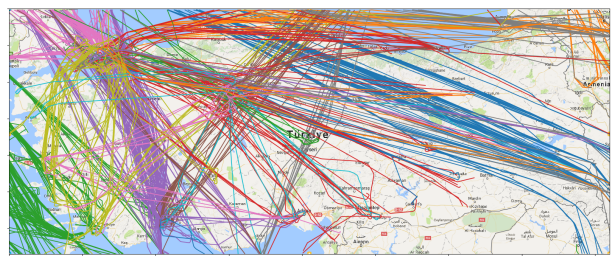

Fig. 10. Clusters formed with DBSCAN

The problem with DBSCAN method is that parameters change result dramatically. One configuration may put every trajectory into one large cluster. Another configuration may not even find a cluster and label every point as noise. Parameters should be chosen carefully but having two parameter makes this tuning difficult. Also, the parameters should be tuned again if number of sample changes. Even though we add the same points to dataset without changing point distribution, DBSCAN results different clusters. This is because cluster densities change.

Given good pair of parameters, DBSCAN performs better when clusters have similar density. However, this is not the case in our dataset. 3 dense trajectory clusters are merged and shown as 1 cluster. If we adjust parameters to have 3 cluster in that region, other clusters are diagnosed as noise.

D. HDBSCAN

DBSCAN fails when cluster densities are very different from each other. HDBSCAN is proposed to solve this problem. HDBSCAN is a hierarchical clustering method. By forming sub clusters, HDBSCAN can detect clusters with lower density [9]. It can detect noise samples as DBCAN does. Also, it scan various ε values and finds clusters which better stability over ε.

However, HDBSCAN reported over 20 percent noise even though we set min cluster size parameter as small as possible

to decrease noise. Trajectories assigned to a cluster are shown blue and noise trajectories are depicted red in figure 11.

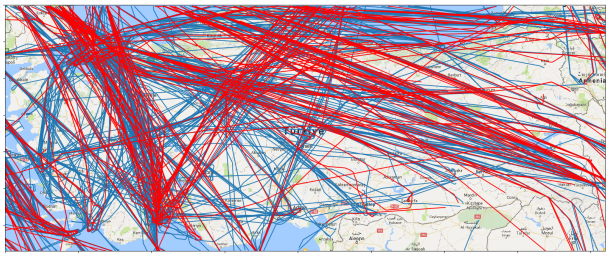

Fig. 11. Most of the trajectories are detected as noise

VI. Results

Silhouette score is used to compare clustering of different methods. It ranges between -1 and 1. Silhouette index reflects compactness and separation of clusters [10]. Higher scores mean better defined clusters. for a single sample, it is defined as:

$$s = \frac{b - a}{max(a, b)}$$

where a is average distance between the sample and all other samples in the same cluster and b is average distance between the sample and other samples in the next nearest cluster. Distance is defined as hausdorff distance. Silhouette score is mean silhouette index of each sample in the dataset. Scores for various method-parameter configuration is given below.

Method	Parameter 1	Parameter 2	Silhouette Score
K-medoids	$K = 20$	-	0.232
GMM	no of Cluster $= 8$	feature-length $= 20$	0.112
GMM	no of Cluster $= 8$	feature-length $= 10$	0.112
GMM	no of Cluster $= 8$	feature-length $= 5$	0.108
GMM	no of Cluster $= 8$	feature-length $= 2$	-0.080
DBSCAN	$\varepsilon = 120000$	min samples $= 50$	0.213
HDBSCAN	min cluster size $= 2$	-	-0.143

It should be noted that these scores are given as a formal basis for comparison. It is not considered as an objective evaluation metric. It has biases towards some methods. For example, what silhouette score calculates is actually objective function of K-medoid algorithm. So we expect K-medoid to have higher score. Apart from that, DBSCAN and K-medoids have higher score than others.

K-medoids is simple implement, faster than other algorithms. It is more intuitive. It has only one parameter and this parameter is easy to tune. We can even adjust it without looking dataset beforehand. DBSCAN also has high silhouette score. It can also detect outliers but right parameters are hard to find and parameters should be adapted if data size changes. HDBSCAN is an alternative to DBSCAN but it performed poorly in our dataset. It didn't give meaningful results. GMM follows a probabilistic approach. It highly depends on feature representation and it needs strong assumptions.

VII. Toolset

Python data science packages (numpy[11], scipy, scikit-learn[12], matplotlib) are used in every step. A couple of k-medoids implementations[13][14] are inspected and adapted for our use case. hdbscan package of python is used in HDBSCAN clustering part[15].

REFERENCES

[1] Laxhammar, Rikard, and Gran Falkman. "Online learning and sequential anomaly detection in trajectories." IEEE transactions on pattern analysis and machine intelligence 36.6 (2014): 1158-1173.

[2] Palma, Andrey Tietbohl, et al. "A clustering-based approach for discovering interesting places in trajectories." Proceedings of the 2008 ACM symposium on Applied computing. ACM, 2008.

[3] Vaughan, Neil, and Bogdan Gabrys. "Comparing and combining time series trajectories using Dynamic Time Warping." Procedia Computer Science 96 (2016): 465-474.

[4] Chen, Zezhi, Yuyao Yan, and Tim Ellis. "Lane detection by trajectory clustering in urban environments." Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014.

[5] Bergroth, Lasse, Harri Hakonen, and Timo Raita. "A survey of longest common subsequence algorithms." String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on. IEEE, 2000.

[6] Chen, Jinyang, et al. "Clustering of trajectories based on Hausdorff distance." Electronics, Communications and Control (ICECC), 2011 International Conference on. IEEE, 2011.

[7] Guo, Yuejun, et al. "Trajectory Shape Analysis and Anomaly Detection Utilizing Information Theory Tools." Entropy 19.7 (2017): 323.

[8] Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.

[9] Campello, Ricardo JGB, Davoud Moulavi, and Jrg Sander. "Density-based clustering based on hierarchical density estimates." Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Berlin, Heidelberg, 2013.

[10] Wang, Kaijun, Baijie Wang, and Liuqing Peng. "CVAP: validation for cluster analyses." Data Science Journal 8 (2009): 88-93.

[11] Computing in Science & Engineering 9, 10 (2007); 10.1109/MCSE.2007.58

[12] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[13] GitHub. (2018). [online] Available at: https://github.com/alexprengere/medoids [Accessed 4 Jan 2018].

[14] GitHub. (2018). [online] Available at: https://github.com/letiantian/kmedoids [Accessed 4 Jan 2018].

[15] L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering In: Journal of Open Source Software, The Open Journal, volume 2, number 11. 2017