Scooter Trajectories Clustering

University of Verona
Computer Engineering for Robotics and Smart Industry
Machine Learning and Deep Learning
2020/2021

Mirco De Marchi - VR445319

***Abstract*—The aim of this review is to present the main unsupervised learning techniques used for scooter trajectories clustering. The dataset that I used contains a big amount of positions taken in rentals that run through some cities in Italy. The objective is to find recurrent places crossed by the trajectories. This review presents some analysis on the positions and the implementation of some heuristics that manage data in a systematic way: the *spreaddelta heuristic*, the *edgedelta heuristic* and the *coorddelta heuristic*. Then I performed some machine learning clustering techniques on the dataset: *mean shift*, *full linkage*, *ward hierarchical* and *k-means*, adding the features extracted from the heuristics performed before. All the features are extracted with *Principal Component Analysis (PCA)* with an improvement that select the component to focus on and the features has been integrated with the informations obtained from the heuristic procedures in order to optimize feature extraction.**

I. Motivation

Motion trajectories are really difficult to analyse and handle because of the amount of data. There could be errors in position tracking and usually you don't have informations like the city from which the positions are taken, or any higher semantic level information that can help in trajectories grouping. Consequently trajectories are difficult to represent, filter and manage in relation with itself or other data. First of all, trajectories clustering is a challenge for its intrinsic difficulty in being treated and today is an ambitious topic in data science research. Moreover trajectory clustering can help for several applications:

- Monitoring: understand main points of interest and common places visited by tourists;
- Forecasting: prediction of possible destinations starting from a position;
- Viability: traffic monitoring and detection of possible point of traffic jam;

II. State of art

In [1] book there are some techniques widely used for clustering of moving object. Some methodology are pretty new but there are also some historical algorithm that provides good results:

- K-Means:
- CURE:
- Birch:
- DBSCAN:
- Optics:
- Sting:
- Cosweb

III. Objectives

The objectives of this project is grouping trajectories in order to find the common location crossed by people that rent a scooter to visit a city in Italy. The model built has to be able to distinguish for example the positions related to a trajectory that takes from the train station to the city centre, the ones that move in the city center, the ones that run through the periphery and so on.

In particular the steps that involves the project are:

1) Dataset filtering and merging in order to build a dataset tidy and semantically correct;
2) Analyse features and represent the trajectories in a line plot;
3) Group the positions for each rental and sort them through the timestamp;
4) Apply heuristic techniques on features;
5) Apply unsupervised learning techniques on features (as K-Means, Mean Shift...);

IV. Methodology

The original dataset is composed by 4 tables in CSV fomat: *pos.csv*, *rental.csv*, *user.csv*, *device.csv*. This dataset is a subset of an another dataset with some sensitive data dropped (as user name or email) and positions manumit in order to protect proprietary data. Although the data are only a subset of the proprietary data, the amount positions is really huge and it weighs about 2GB.

The *pos.csv* table contains all the positions, characterized by latitude, longitude, speed, timestamp and a device id used to join with *device.csv* table. The *device.csv* and *user.csv* tables contain the kilometers traveled respectively by a scooter and by a user. At last, the *rental.csv* table contains the longitude and latitude positions of rental starting and ending points with related timestamp, and all the ids used to join the other tables with this one.

The first thing that I have noticed is that the dataset is not built very well, because the informations referenced are not semantically correct. In fact, I would have the positions in relation with the rental and not with the device, therefore I tried to join the tables and I noticed that each rental is referenced by an unique device id. It means that there not exist multiple device used in different rental, but the device table is used

only to take the positions in relation with rentals. Moreover I noticed that not all positions data have a matched rental, and not all rentals have some matched positions. This is caused by the subset extraction and the previous operation performed to protect sensitive informations. It means that the dataset can be filtered and reduced in size.

As result of this first analysis I implemented an algorithm that filters and joins the table in a single one, in order to have simpler data to handle. This operation was really tricky because the total amount of data is huge and I had to use optimized functions based on database join algorithms and chunks management to be able to handle these data in shot time. At last, the resulting table has been sorted for rental id, position id and timestamp, in order to have everything ready to perform plot representation and some feature analysis.

After that I performed some analysis on the features. I studied their distribution and I started to think how could be possible to group or divide positions. In particular I learnt that two trajectories can be similar in shape and can be divided in time. It means that a sequence of positions, a trajectory, can be similar to another one, evaluating his shape appearance and location in the geographical map, and therefore his sequence of latitude and longitude. On the other hand a trajectory can be different to another one if there is a temporal space between each other. The starting assumption of this analysis is that all positions belong to a rental, it means that starting from how data are constructed, I assume that a trajectory is the sequence of positions sorted in time that belong to the same rental. As result I implemented 3 different clustering heuristics performed in a systematic and statistical way on the entire sequence of positions:

- **timedelta heuristic**: considers that a trajectory of a rental can be divided in a sequence of trajectories if the time gap between a position and previous one exceeds a *timedelta* value. First of all I calculated the time gaps for each set of positions grouping in rental:

$$TIMEGAPS = \{p.time - p[-1].time \mid \forall p \in POS\} \quad (1)$$

  where `p[-1]` is the previous position and the field `.time` is the related timestamp. The *timedelta* value can be assigned by a user that runs the heuristic process or can be automatically calculated. To calculate automatically the timedelta value I plotted the timegaps distribution, I noticed that the curve is unimodal and I assumed to approximate it as a normal distribution. Therefore I assigned the timegaps standard deviation to my timedelta in order to exploit the statistical empirical rule and cover a percentage of timegaps distribution. In this case I take all the left tail of the distribution and the 43% of the right one, and that ones that remains outside are the positions candidates that divide a trajectory from another one. This operation has been performed for each rental positions in order to obtain a set of sub-trajectories of rental trajectory.
- **spreaddelta heuristic**: considers that a trajectory of a rental can be considered similar to another one if they spread the same area. I calculate the spread area for each rental trajectory in the following way:

$$SPREADS = \{max(t) - min(t) \mid \forall t \in TRAJ\} \quad (2)$$

  where `max(t)` and `min(t)` calculate respectively the maximum and the minimum latitude and longitude of a set of positions, and `TRAJ` is the set of trajectories that can be grouped for each rental, but even for timedelta heuristic previously calculated in order to consider the time gaps division of trajectory and not only the rental division. Also in this case the result of spread distribution is a unimodal curve that can be approximated in a normal distribution and I have again exploited the empirical rule to compute the trajectories similar for spread. I assigned $std(SPREADS)/4$ to the spreaddelta if you want to automatically calculate it, otherwise of course you can choose the value of spreaddelta. If automatically calculated, the spreaddelta will cover the spread distribution of about 20%. This operation will be repeated as long as all the positions has spread id assigned, and when a position is assigned it is discarded from the distribution calculus.
- **edgedelta heuristic**: acts as the spreaddelta heuristic, but it consider the edge of a trajectory, or rather the first position and the last position of the trajectory. The main problem here is that the distribution of edge positions has 2 centres or, in other words, it is bimodal. In fact the dataset positions involves mainly in 2 different cities of Italy, and this translates in a bimodal distribution of the positions. In this case I can't use the distribution mean or the standard deviation because they should result wrong, but I have to approximate these values. To resolve this issue I applied for simplicity only 1 iteration of Mean Shift starting from a random position in order to get closer to a centre of the distribution (one of the two indifferently) and to obtain a spread subset more significant to calculate the deviation standard and use it as edgedelta. In particular the value assign to edgedelta is $std(meanshift(EDGES, iter = 1)) * 2$ and obtain the 95% of the subset of edges found by meanshift. The set of edges is calculated in the following way:

$$EDGES = \{concat(p[0], p[-1]) \mid \forall t \in TRAJ\} \quad (3)$$

- **coorddelta heuristic**: it is a combination spread and edge heuristics and combine the main advantages of each other. To guarantee the convergence of this algorithm edge heuristic and spread heuristic have to manage the same subset of trajectory. Therefore first it is applied edge heuristic to find a subset of trajectories near a distribution centre and then both heuristics, edge and spread, are applied to same subset.

After the implementation of these heuristic techniques, I started to prepare the features to be processed by clustering algorithms. In particular I decided to perform Standardization, Normalization and than Principal Component Analysis (PCA) on features. The component extracted by PCA can be dediced in 3 different ways:

1) By a number of component decided a priori;

2) By the cumulative variance calculated by PCA over the number of features, considering to cover almost an 80% of variance;
3) Constructing a list of features subset and performing PCA to produce 1 component for each features subset. The result will be a concatenation of columns produced by PCA for different subset of features.

As feature extraction, I considered that the work done for the heuristic algorithm could return useful for clustering algorithms. In particular $TIMEGAPS$, $SPREADS$ and $EDGES$ sets can be used as new features of my data. Therefore I integrate my data features with the ones extracted from the heuristic and I used it to be processed with Standardization, Normalization, PCA and than clustering algorithms.

The clustering algorithms that I used are the following:

- **K-Means**:
- **Mean Shift**:
- **Gaussian Mixture**:
- **Full Hierarchy**:
- **Ward Hierarchy**:

V. Results

The result of this clustering analysis I want to highlight 2 main aspects:

VI. Conclusion

References

[1] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, 01 2017.

Figure 1. Original dataset ER model

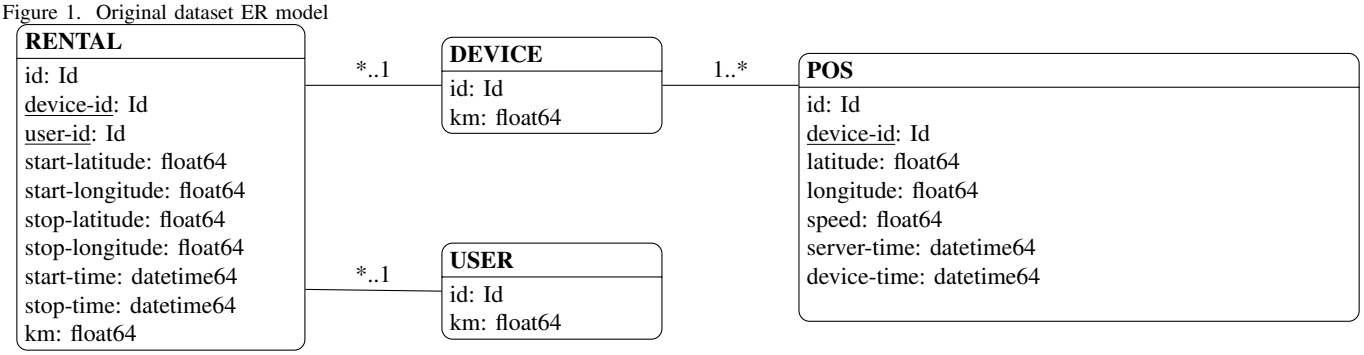

Figure 2. Generated dataset ER model