MovinPy – Moving Objects' Data Preprocessing Library in Python

George S. Theodoropoulos, Yannis Theodoridis

Data Science Lab., University of Piraeus

vers. 1.0

1. Introduction - Project Goals

The MovinPy project aims at creating a python library that can be used to process and analyze mobility data. MovinPy library includes a set of well documented and user friendly functions, since ease of use and expandability are top priorities of this project. The objective is that upon project completion, the end user will be able to use this library to clean, transform and analyse trajectory data, a process that will enhance his/her understanding of the dataset at hand and prepare the dataset as input for further processing and analytics / machine learning tasks (classification / prediction, etc.).

The library is developed using Python 3.

2. Library high-level description

The library includes the following components:

Group A: raw data preprocessing and trajectory reconstruction

1. Data cleansing.
2. Point of Interest (PoI) discovery and semantic enrichment.
3. Trajectory reconstruction (incl, gap detection).
4. Trajectory resampling
5. Trajectory simplification/compression.

Group B: statistical analysis for data understanding

1. Signal-level statistical analysis.
2. Trajectory-level statistical analysis.

Group C (misc)

1. Data stream processing (adapting some of the Task A methods for use in a streaming environment.

3. The library in detail

Hereafter, the input / output files processed by the methods in the library will conform to one of the X types below (denoted, resp., as [input|output]-type1.csv, …, [input|output]-typeX.csv in the method declaration):

- type1- input/output: (oid, ts, [lat, lon/geometry])
- type2- input/output: (oid, ts, [lat, lon/geometry], tid)
- type3- input/output: (oid, ts, lat, lon, tid)

where oid is the unique moving object identifier, tid is the moving object's trajectory identifier (makes sense only after the trajectory reconstruction process, see A.3 below), ts is the timestamp of the signal, and (lat, lon) is the pair of spatial coordinates of the signal.

Other assumptions:

- Input files are assumed to be sorted by (ts, oid[, tid])
- Unless specifically declared, no assumption (e.g. linear interpolation) is applied for the movement in between two successive locations of an object.

GROUP A

A.1 - Data cleansing and outlier detection

Data cleansing consists of two tasks:

1. Duplicates cleansing
2. Outlier detection and cleansing based on thresholds/quantile outliers on time, spatial, and speed features

cleanse_and_detect_outliers.py input_type1/2.csv -o output.csv --outliers=["drop", "return"] --features=columns_with_outliers --alpha=iqr_score_alpha

Cleanse.py cleans the input csv from (a) duplicate records and, optionally, (b) removes or returns outlier values on specific columns. In particular for (a), two records are considered duplicate if two records have identical object ids (oid) and timestamps (ts). In particular for (b), the method used to detect outlier records is the IQR scoring method (https://en.wikipedia.org/wiki/Interquartile_range). This method detects the records that exist outside a quantile range that is picked based on the variance of the data and the alpha value picked (default alpha=3, the larger the alpha, the wider the range of acceptable values) to identify records that are considered noise/error.

A.2 - Point of Interest (PoI) tagging

Point of Interest tagging is the process that enriches each record with information regarding nearby predefined Points of Interest.

PoI_tagging.py input_type1.csv pois_type1/2.csv -o output.csv --epsg=projection --threshold=Kilometers --tag_column=column

Tagging each record with respect to its proximity to a set of POIs supplied by the user (pairs of lat/lon). A new column (named "poi_id") is added to the original input table. This column contains the identifier of the POIs that are at maximum "threshold" Kilometers away from the record. If the "tag_column" argument is not used, the index of the POI is used as the "poi_id" identifier. If a specific column is set as the "tag_column", then the record's attribute at the specified column is used as the "poi_id" identifier. An appropriate epsg needs to be supplied (one that has meters as a metric). The supplied coordinate data (lat,lon) will be transformed using the supplied epsg, thus one tailored to the given dataset will supply more accurate results.

A.3 - Trajectory reconstruction /Data segmentation (incl. Gap detection)

Trajectory reconstruction is the process of splitting the sequence of signals corresponding to a moving object into distinct subsequences that make sense (called, trajectories). Timestamp gap detection is used as the metric that signals the end of a trajectory and the start of a new one.

Construct_trajectories_using_gaps.py input_type1.csv -o output.csv --threshold=time_threshold

Gap_Detection scans the per object ts column in order to detect gaps in sampling (time differences of more than the threshold specified, e.g. 6 hours). Then, the indices that are found by the gap_detection algorithm are piped to the segmentation algorithm in order for separate trajectories to be formed. A column named "tid" is used to signal different trajectories. This column contains the integer identifier for each unique object.

A.4 - Trajectory resampling

Trajectory resampling is performed as follows:

- Data is grouped using the combination of the 'oid' and 'tid' columns, in order to group the records in distinct trajectories for each object

- The ts column is transformed to one with a constant sampling rate (specified by the user).
- The rest of the Dataframe is filled with Nan values for each record that is new record (the records that don't match the original timestamp).
- For each new record (with NaN), the values of each column are interpolated using the original data.

The resampling step is vital in order for the resulting dataset to be able to be used as input for ML/DL algorithms.

Trajectory_Resampling.py input_type2.csv -o output.csv --sampling_rate

Trajectory_Resampling resamples each unique trajectory of each object using the rule specified (a time interval). The rest of the features are interpolated on the resampled records.

A.5 - Trajectory simplification/compression

A python implementation of the Meratnia and De By td-tr compression algorithm. (https://link.springer.com/chapter/10.1007/978-3-540-24741-8_44)

Trajectory_Compression.py input_type1/2.csv -o output.csv --threshold=time threshold (ex. "6h")

GROUP B

B.1/B.2 - Signal level/Trajectory level statistical analysis

Implementation of methods that return a set of statistical metrics and plots for the given data, both pre and post trajectory creation (depending on the mode selected).

B.1 - Signal level statistical analysis

Stats.py input_type3.csv -o output.csv --mode=record --feature=column --bins

Metrics include:
1. Sample size
2. Sample min/max value
3. Sample mean
4. Sample variance
5. Sample skewness
6. Sample kurtosis

Plots include:

1. Feature Probability Density Function
2. Feature boxplot

B.2 - Trajectory level statistical analysis

```
Stats.py input_type3.csv  -o output.csv --mode=trajectory --bins
```

Metrics include:

1. Number of unique trajectories
2. Mean number of trajectories per object
3. Mean time duration per trajectory
4. Mean first-last point distance per trajectory

Plots include:

1. Trajectories per object Probability Density Function
2. Trajectory duration Probability Density Function
3. First-Last point distance Probability Density Function

4. The library in action

The folder **screenshots** contains screenshots of runs of each module that is part of the MovinPy library. It also contains screenshots of the helper module init_df.py that is used to initialize a dataset (convert it to type1), two sample figures produced by the stats.py script and a screenshot of pip freeze of the modules environment. The trial runs were based on the Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance.