Mobility Patterns
Content of this lesson

- Global patterns
  - Trajectory distances
  - Trajectory clustering
- Local patterns
  - Flocks, Convoys & Swarms
  - Moving clusters
  - T-Patterns
Global Patterns
Clustering
(sample K-means family)

- Find k subgroups that form compact and well-separated clusters
Trajectory clustering

- Trajectories are grouped based on similarity


Trajectory Clustering

Questions:
- Which distance between trajectories?
- Which kind of clustering?
- What is a cluster ‘mean’ in our case?
  - A representative trajectory?
Trajectory Distances
Families of Trajectory Distances

- Trajectory as **set** of points
  - Single-point approaches
  - Hausdorff distance
- Trajectory as **sequence** of points
  - Fréchet distance
  - Time series distances: Euclidean, DTW & LCSS
- Trajectory as **time-stamped sequence** of points
  - Average Euclidean distance
Trajectory as set of points
Common Destination

- Select last point $Plast$ for each trajectory
- $D(T, T') = \text{Euclidean}(Plast, P'last)$
Select first point $P_{\text{first}}$ for each trajectory

$D(T,T') = \text{Euclidean}(P_{\text{first}}, P'_{\text{first}})$
Trajectory as set of points

Hausdorff distance

- Intuition: two sets are close if every point of either set is close to some point of the other set
- Formally, given sets $A$ and $B$:
  - $r(x, B) = \inf \{d(x, b) : b \in B\}$
  - $h(A, B) = \sup \{r(a, B) : a \in A\}$
  - $d_H(A, B) = \max \{h(A, B), h(B, A)\}$

- Equivalently:
  - $h(A, B) = \text{minimum buffer radius around } B \text{ that fully contains } A$
  - $d_H(A, B) = \text{symmetric version of } h()$
Trajectory as set of points
Hausdorff distance

Example $h(A,B)$
Trajectory as sequence of points
From Hausdorff to Fréchet distance

- Applied to trajectories, sometimes Hausdorff distance yields counter-intuitive results
- How far are these?

- Reasonable in a set-oriented view
- Wrong in terms of moving objects
Trajectory as sequence of points
Fréchet distance

- Intuition: equivalent of Dynamic Time Warping on continuous curves
- Formally:

\[ F(A, B) = \inf_{\alpha, \beta} \max_{t \in [0,1]} \left\{ d\left( A(\alpha(t)), B(\beta(t)) \right) \right\} \]

\(\alpha\) and \(\beta\) are non-decreasing mappings from \([0,1]\) to the points along \(A\) and \(B\) in forward order

- Also described as “minimum leash length”:
  - What is the minimum length of a leash needed to stroll around the dog, given the owner’s and the dog’s trajectories?
Trajectory as sequence of points
Fréchet distance

- Back to our example
Trajectory as sequence of points
Time series distances

- Just replace “difference of two values” with “spatial distance of two points”
- IMPORTANT: most methods in this class assume constant sampling rates

- Examples:
  - Dynamic Time Warping
  - Edit Distance with Real values
    - Similar to DTW, but can remove points

Dynamic Time Warping Matching
Longest Common SubSequence
- Define a maximum radius
- Match points from the two trajectories if dist() < radius
- Find contiguous subsequences of matches
- LCSS = length of the best match
The trajectory is seen as a continuous spatio-temporal curve
Positions between input points (the GPS fixes) linearly interpolated

"Synchronized" behaviour distance
  Similar objects = almost always in the same place at the same time
Computed on the whole trajectory

\[
D(\tau_1, \tau_2) = \frac{\int_{T} d(\tau_1(t), \tau_2(t)) dt}{|T|}
\]

distance between moving objects \(\tau_1\) and \(\tau_2\) at time \(t\)
Clustering Algorithms
Which kind of clustering method?

- In principle, any distance-based algorithm
- General requirements:
  - Non-spherical clusters should be allowed
    - E.g.: A traffic jam along a road = “snake-shaped” cluster
  - Tolerance to noise
  - Low computational cost
  - Applicability to complex, possibly non-vectorial data
- A suitable candidate: Density-based clustering
  - OPTICS (Ankerst et al., 1999)
  - Evolution of standard DBSCAN
Density Based Clustering
A refresher

K-means

Density-based

cluster 1
cluster 2
cluster 3
cluster 4
Density Based Clustering

Step 1: label points as core (dense), border and noise

- Based on thresholds R (radius of neighborhood) and min_pts (min number of neighbors)
Density Based Clustering

Step 2: connect core objects that are neighbors, and put them in the same cluster
Density Based Clustering

Step 3: associate border objects to (one of) their core(s), and remove noise
Density Based Clustering

Original Points

Point types: core, border and noise
Density Based Clustering

- Resistant to Noise
- Can handle clusters of different shapes and sizes
A sample dataset

- A set of trajectories forming 4 clusters + noise (synthetic)
T-OPTICS vs. K-means

Reachability plot
(= objects reordering for distance distribution)
What’s the source of traffic in Pisa?

Trajectory clustering at work
Access patterns using T-clustering
Characterizing the access patterns: origin & time
Local Trajectory Patterns
Frequent patterns in sequences

• Frequent sequences (a.k.a. Sequential patterns)
• Input: sequences of events (or of groups)
From trajectories to sequential patterns: the easy way

- Map each trajectory to a sequence of areas
  - Predefined or driven by data

<table>
<thead>
<tr>
<th>O</th>
<th>N</th>
<th>D</th>
<th>C</th>
<th>G</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>E</td>
<td>H</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

A → B → C

D → B → E → F

C → G → H → E → I

L → M → H
From trajectories to sequential patterns: the easy way

- A “Trajectory frequent pattern” can be defined as sequential pattern over traversed areas
Moving Trajectory Flocks

- Group of objects that move together (close to each other) for a time interval
Moving Trajectory Flocks

- Group of objects that move together (close to each other) for a time interval

- Discover all possible:
  - sets of objects $O$, with $|O| > \text{min\_size}$ and
  - time intervals $T$, with $|T| > \text{min\_duration}$
  - such that for all timestamps $t \in T$ the points in $O|t$ are contained in a circle of radius $r$

Moving Trajectory Flocks
From Flocks to Convoys

- Given radius $r$, size $m$, and time threshold $k$
  - find all groups of objects so that each group consists of density-connected objects w.r.t. $r$ and $m$
  - during at least $k$ consecutive time points
- Basically replace circles with DBSCAN clusters
From Convoys to Swarms

- Given radius $r$, size $m$, and time threshold $k$
  - find all groups of objects so that each group consists of **density-connected objects** w.r.t. $r$ and $m$
  - during at least $k$ time points – **not necessarily** consecutive

swarm pattern = \{O_1, O_2, O_3, O_4\} over times \langle 1, 3 \rangle
Moving Clusters

- A **moving cluster** is a set of objects that move close to each other for a long time interval.

Formal Definition [Kalnis et al., SSTD’05]:
- A **moving cluster** is a sequence of (snapshot) clusters $c_1, c_2, \ldots, c_k$ such that for each timestamp $i$ ($1 \leq i < k$): $\text{Jaccard}(c_i, c_{i+1}) \geq \theta$
  - $\text{Jaccard}(c_i, c_{i+1}) = \frac{|c_i \cap c_{i+1}|}{|c_i \cup c_{i+1}|}$
  - $0 < \theta \leq 1$
- Clustering computed with density-based method (DBSCAN)
Moving Clusters

[Diagram showing clusters of sheep moving over time with annotations 75% and OK]
T-Patterns

- A sequence of visited regions, **frequently** visited in the specified order with similar transition times

\[ A_0 \xrightarrow{t_1} A_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} A_n \]

\( t_i = \) transition time, \( A_i = \) spatial region

T-Patterns

Key features
- Includes typical transition times in the output
- Areas are automatically detected – not “the easy way”
Sample Trajectory Pattern

Data Source: Trucks in Athens (273 trajectories)

A → B → B and
A → B' → B''
A quick peek into
Deep Learning
Deep Learning approaches

- Sample approach: DETECT: Deep Trajectory Clustering for Mobility-Behavior Analysis
- Basic idea:
  - Trajectories
  - Embedding
  - K-means
- Integrate the clustering step in the learning of embeddings
- Three steps:
  - Enrich trajectories with context
  - LSTM-based embedding of trajectories
  - Clustering on embeddings

Enrich trajectories with context
- Identify stay areas = segment of trajectory where there is no movement, basically a stop
- Create a buffer around the area
- Select all points-of-interest located there (hotels, shops, etc.)
- Compute a feature vector, one feature per PoI category

Output
- Traj = < (x,y,[f₁,..., fₙ]), (x',y',[f'₁,..., f'_ₙ]), ... >
LSTM-based embedding of trajectories
  - Apply a encoder-decoder schema to the enriched trajectories
  - Use LSTM as basic mechanism

Objective: minimize the difference between the encoder input and the decoder output
LSTM-based embedding of trajectories
  ○ Apply an encoder-decoder schema to the enriched trajectories
  ○ Use LSTM as the basic mechanism

Objective: minimize the difference between the encoder input and the decoder output
- Clustering on embeddings
- Clustering error becomes one term of the overall loss function
- P & Q = points distribution
  - P = real data (embedded)
  - Q = clusters (Student t-distribution around centers)

\[
\ell_c = KL(P \| Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}
\]

\[
\ell_r = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T_i} \sum_{t=1}^{T_i} (x_i^{(t)} - \hat{x}_i^{(t)})^2
\]

\[
\nabla_c
\]

\[
\nabla_r
\]

Standard auto-encoder cycle
Homeworks
Homework 7.1

Implement a simple (non-optimized) discrete version of Hausdorff distance for trajectories, i.e. considering only the GPS points and not the segments connecting them:

- Apply it to a set of taxi trips: randomly pick 10 trajectories as “query objects”; find for each of them the trips of the dataset having $d_H(.) < 500$ mt; show them (query + result) on the map.
- Write a (well commented) python notebook, where $d_H$ is defined as a function
Define a simple “embedding” of trajectories, e.g. as trajectory length, main direction, average latitude, etc. (you decide the number of features to use); then cluster the embeddings (you decide the clustering algorithm); finally, show on a map the different clusters.

- Apply it to a (sub)set of taxi trips, e.g. SF.
- Write a (well commented) python notebook
Homework 7.3

Clustering trajectory segments (a.k.a. mimicking TraClus [1])

Strongly simplify a dataset $D$ of trajectories (output = $D'$), then build a second dataset $D''$ containing, for each trip in $D'$, all its segments. Then, cluster the segments in $D''$ using the coordinates of start and end as attributes for clustering (4 attributes per segment), and show results on a map. You decide the clustering algorithm to use.

- Apply it to a (sub)set of taxi trips, e.g. SF.
- Write a (well commented) python notebook

[1] https://pypi.org/project/traclus-python/