COMPSCI 531D Fall 2021
Project 1: Trajectory Data Analysis

Due Date: September 28, 2021

1 Introduction

This project asks to design, implement, and analyze algorithms to perform various tasks for trajectory data analysis. A trajectory is a curve representing the change of a set of variables over time. Trajectory data analysis arises in a range of applications, including traffic data analysis, analyzing migration and commute patterns, computational finance, computational molecular biology, computer vision, robotics, and machine learning. This project focuses on the task of summarizing large trajectory datasets with a few representative trajectories.

The project is composed of four components. The first three components ask to design and implement algorithms to *cluster* the trajectories using subroutines partially described in this document, perform experimental analyses, and discuss possible improvements. The fourth component asks for extending algorithms in various directions, such as scalability of the algorithms, handling noisy and missing data, and application-specific necessities such as fast updates to the dataset.

All implementations will be submitted as source code in an approved programming language, and all analyses, figures, and discussions will be included on presentation slides; see Section 7 for more details.

2 Problem Formulation

Formally, a trajectory is the image of a function $T : [0,1] \to \mathbb{R}^d$, which we also denote by T. Assume $d = 2$ throughout this project. T is represented as a sequence of points sampled on the curve, i.e., $T = \langle p_1, p_2, \ldots, p_m \rangle$, where $p_i = (a_i, b_i) \in \mathbb{R}^2$ is a point in the plane. For simplicity, assume the actual curve is obtained by performing linear interpolation between two consecutive sample points, i.e., T is assumed to be a polygonal curve with vertices $p_1, p_2, \ldots, p_m$. See Figure 1. Let $|T|$ denote the number of vertices in T, and let $\|T\| = \sum_{i=1}^{m-1} \|p_i - p_{i+1}\|$ be its arc length, where $\|p_i - p_{i+1}\| = \sqrt{(a_i - a_{i+1})^2 + (b_i - b_{i+1})^2}$ is the Euclidean distance between p_i and p_{i+1}. Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of trajectories. Set $N = \sum_{i=1}^{n} |T_i|$ to be the total input size. For a pair of trajectories S, T, let $\Delta(S, T)$ be a *distance function* that measures the distance between S, T (see Section 3 for a couple of examples).

A *k-clustering* of $\mathcal{T}$ is a partition $\mathcal{C} = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ of $\mathcal{T}$ into k pairwise-disjoint subsets. For each cluster $\mathcal{T}_i$, let C_i be a *center* trajectory of $\mathcal{T}_i$, defined as

$$C_i = \arg\min_{T} \sum_{T' \in \mathcal{T}_i} \Delta(T, T'),$$

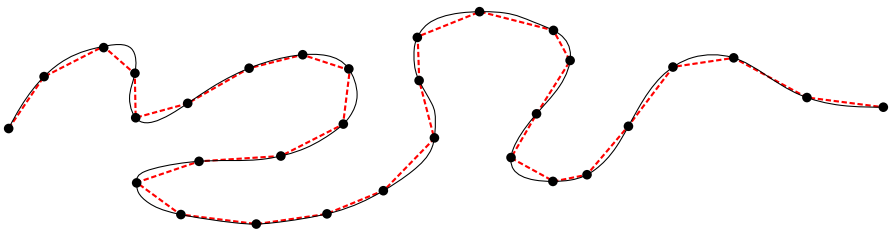


Figure 1: An original trajectory and its sampled points. The dashed trajectory is obtained by linearly interpolating between consecutive points.

where T is a trajectory represented as a sequence of points. Set

$$\$(\mathscr{T}_i, C_i) = \sum_{T' \in \mathscr{T}_i} \Delta(C_i, T').$$

Define the *cost* of $\mathscr{C}$ to be $\$(\mathscr{C}) = \sum_{i=1}^{k} \$(\mathscr{T}_i, C_i)$. Ideally, the goal is to compute a k-clustering $\mathscr{C}^* = \arg\min_{\mathscr{C}} \$(\mathscr{C})$, where the minimum is taken over all k-clusterings of $\mathscr{T}$. Let $\mathscr{C}^* = \{\mathscr{T}_1^*, \ldots, \mathscr{T}_k^*\}$ be an optimal k-clustering of $\mathscr{T}$ with C_i^* being the center trajectory of T_i^*. Then for all $1 \le i \le k$ and for all $T \in \mathscr{T}_i^*$, $\Delta(C_i^*, T) \le \Delta(C_j^*, T)$ for all $j \le k$.

Computing a minimum-cost k-clustering, or even computing a center trajectory, is computationally intractable, so the goal will be to efficiently compute a k-clustering of small cost.

3 Part 1: Comparing Trajectories

Measuring the distance between a pair of trajectories is a well-studied task on its own, with many proposed solutions. This project asks to implement two closely related distance measures and compare them empirically.

Both distance measures below are defined with respect to *correspondences*. Let $T_1 = \langle p_1, \ldots, p_r \rangle$ and $T_2 = \langle q_1, \ldots, q_s \rangle$ be trajectories represented by two sequences of points in $\mathbb{R}^2$. A correspondence is an ordered pair of the form (p_i, q_j). A set C of correspondences is *monotone* if for any two correspondences $(p_i, q_j), (p_{i'}, q_{j'}) \in C$ with $i < i'$, we have $j \le j'$. In other words, the pairs of correspondences in C do not "cross"; see Figure 2. Furthermore, C is called an *assignment* if every p_i, for $1 \le i \le r$, and every q_j, for every $1 \le j \le s$, appear in at least one pair of C. The *dynamic time warping* (DTW) cost of a monotone assignment C is

$$\mathsf{dtw}(T_1, T_2, C) = \sum_{(p,q) \in C} \|p - q\|^2$$

and the *discrete Fréchet distance* (DFD) cost of C is

$$\mathsf{dfd}(T_1, T_2, C) = \max_{(p,q) \in C} \|p - q\|^2.$$

Define $\mathsf{dtw}(T_1, T_2) = \min_C \mathsf{dtw}(T_1, T_2, C)$, where the minimum is taken over all monotone assignments, and similarlly define $\mathsf{dfd}(T_1, T_2) = \min_C \mathsf{dfd}(T_1, T_2, C)$.

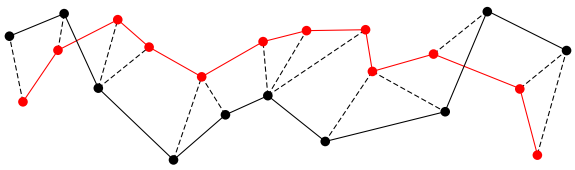


Figure 2: Example of a monotone assignment between two trajectories.

3.1 What to do (Code)

1. Implement a $O(|T_1| \cdot |T_2|)$-time procedure to compute $\mathsf{dtw}(T_1, T_2)$, where T_1, T_2 are trajectories.
2. Implement a $O(|T_1| \cdot |T_2|)$-time procedure to compute $\mathsf{dfd}(T_1, T_2)$, where T_1, T_2 are trajectories.
3. Copy the two procedures above, then modify them to not only return the minimum cost, but also a monotone assignment with the minimum cost. The runtime should remain $O(|T_1| \cdot |T_2|)$.

3.2 What to do (Slides)

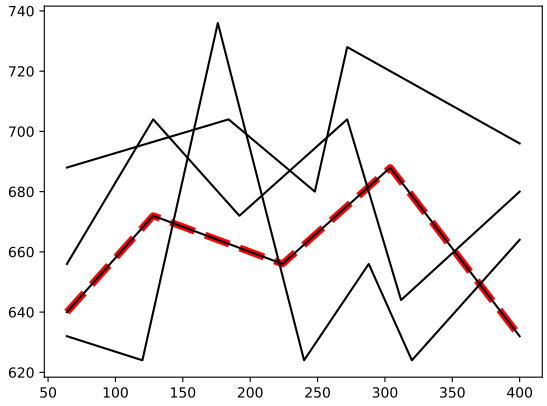
1. Select two trajectories T, T' from the `highway.csv` dataset in lane 1 (see Section 9). Using the previous step, compute $\mathsf{dtw}(T, T')$ and $\mathsf{dfd}(T, T')$. Then repeat this for another pair of such trajectories.
2. Generate figures that illustrate the four monotone assignments computed in the previous step: For each distance function and each pair of trajectories, draw the trajectories and line segments between each correspondence in one image. Include the trajectory IDs used and the costs of the monotone assignments with the figures, e.g. in the captions.
3. Compare and contrast these distance functions, dtw and dfd, by conducting experiments. You may use our provided datasets or generate your own; see Section 10. Include your empirical data and analysis in the slides. Describe the strengths and weaknesses of these distance functions and the algorithms to compute them. Propose alternative measures or extensions to these two, possibly for specific applications, settings, or input trajectories with specific properties.

4 Part 2: Center Trajectories

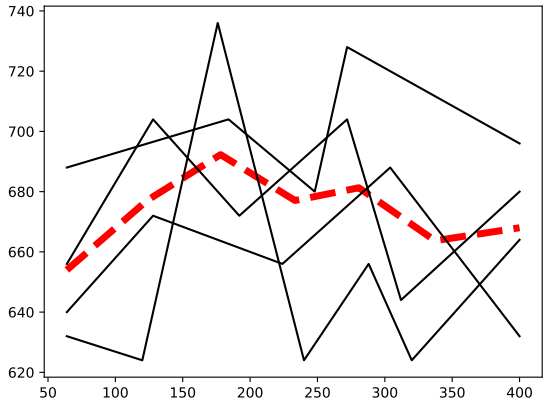
Suppose we have a set X of objects, and a (pairwise) distance function $\Delta : X \times X \to \mathbb{R}$, and we want to compute ***exactly one*** object that "best" represents X with respect to Δ, which is called the *center* of X. Note that the center of X need not be an element of X. This project asks to implement two methods to compute centers of trajectories, then perform experiments to compare and contrast them.

4.1 What to do (Code)

1. Implement a procedure that takes a set $\mathcal{T}$ of trajectories and distance function $\Delta : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ as input and returns the center trajectory of $\mathcal{T}$, $\operatorname{argmin}_{T \in \mathcal{T}} \sum_{T' \in \mathcal{T}} \Delta(T, T')$, in $O(N^2)$ time, where N is the total input size by number of points. See Figure 3(a). In this case, the center trajectory is one of the input trajectories.



(a) center trajectory



(b) 6-average trajectory

Figure 3: Examples of the two types of centers (red) for four input trajectories (black)

arc

2. Recall that, while a trajectory T is represented by a sequence of points in the plane, we use linear interpolation so that T is a polygonal curve. For any $x \in [0,1]$, define $T(x)$ to be the point on T such that the arc length of the initial portion of T until $T(x)$ is $x \cdot \|T\|$.

   Implement a procedure that takes $\mathscr{T}$ and an integer $h > 0$ as input and returns the *h-average* trajectory $C_{\mathscr{T},h}$ of $\mathscr{T}$ of the form $\langle a_0, \ldots, a_h \rangle$, where

$$a_i = \frac{1}{|\mathscr{T}|} \sum_{T \in \mathscr{T}} T\left(\frac{i}{h}\right) \in \mathbb{R}^2,$$

   i.e., a_i is the center of mass of the points $\left\{T\left(\frac{i}{h}\right) \mid T \in \mathscr{T}\right\}$. See Figure 3(b). Note that, unlike the center trajectory described in the previous step, $C_{\mathscr{T},h}$ may not be in $\mathscr{T}$.

4.2 What to do (Slides)

1. Run the two procedures from the previous steps with the trajectories from lane 1 in the `highway.csv` dataset as input. Visualize the computed center trajectories against the input.

2. Run the two procedures from the previous steps with the trajectories from lanes 3 and 4 in the `highway.csv` dataset as input. Visualize the computed center trajectories against the input.

3. Run the two procedures from the previous steps with the trajectories from lanes 4 and 5 in the `highway.csv` dataset as input. Visualize the computed center trajectories against the input. What differences are there with this output compared with that from the previous step?

4. Compare and contrast these center trajectory definitions, and computations of them, by conducting experiments. You may use our provided datasets or generate your own; see Section 10. Include your empirical data and analysis in your slides. Describe the strengths and weaknesses of these center trajectory definitions and your algorithms to compute them. Propose

alternative definitions or extensions of these, possibly for specific applications, settings, or input trajectories with specific properties.

5 Part 3: Clustering Trajectories

Clustering allows us to identify few representatives of a larger data set by the *centers* of those clusters. The project asks to implement one such method in this part, namely Lloyd's Algorithm, as follows. Its input is a set X of objects, a distance function $\Delta : X \times X \to \mathbb{R}$, and integers $k, r, t_{\max} > 0$.

For $1 \leq i \leq r$, repeat the following steps:

1. Randomly partition X into k sets $X_1, \ldots, X_k$.
2. Iteratively repeat the following two steps,

   Center computation. For each $i \leq k$, compute the *center* C_i of X_i.

   Re-assignment. For each object $x \in X$ assign it to the closest center, i.e., set $X_i = \{x \in X \mid \Delta(C_i, x) \leq \Delta(C_j, x)\ \forall j \leq k\}$.

   until either the clustering is the same before and after an iteration or $t_{\max}$ iterations have been performed.
3. Set $\mathscr{C}_i = \{X_1, \ldots, X_k\}$.

Among $\mathscr{C}_1, \ldots, \mathscr{C}_r$, return C_j where $j = \arg\min_{1 \leq i \leq r} \$(\mathscr{C}_i)$.

5.1 What to do (Code)

1. Implement Lloyd's algorithm using Part 1 and Part 2, where the input set of objects is a set of trajectories. Your implementation should allow for the iterations to be performed on-demand so that you have access to the clusters and their centers between each iteration, e.g. in order to monitor the quality of the clusters as more iterations are performed. Ideally, your implemention will also be written so that arbitrary distance and center-computing procedures can be specified as input, but that is not required (make multiple implementations instead).

5.2 What to do (Slides)

1. Run the algorithm with inputs $k = 8, r = 3, t_{\max} = 20$ on the entire `highway.csv` dataset using the DTW distance function from Part 1 and the first center computation from Part 2. Plot the trajectories and color them by the cluster assigned by the procedure. Then repeat this step using the second center computation from Part 2 with suitable choices for h.
2. Set $r = 10, t_{\max} = 20$. For each k from 1 to 16, run the algorithm with inputs $k, r, t_{\max}$ on the `energy.csv` dataset using the DTW distance function from Part 1 and first center computation from Part 2. Plot the costs of the output k-clusterings as a function of k. For $k = 8$, run the procedure again and plot the trajectories and color them by the cluster assigned by the procedure. Then repeat this step using the second center computation from Part 2 with suitable choices for h.

3. Propose potential changes to this very simple algorithm that could lead to computing lower-cost solutions or improved running time without substantially adversely affecting the cost of the solutions.

6 Part 4: Extensions

Parts 1-3 together describe an algorithm to cluster trajectories. Propose ***two*** possible extensions for the algorithm, then implement modifications and perform an experimental analysis to see the impact of your changes on running time, quality of the solution, etc.

Below are some examples that might motivate modifications:

1. Dynamic time warping and discrete Fréchet distance are only two of many distance functions. What other distance functions could be used in Part A, e.g., that is robust under noise, detours, non-uniform sampling, and incomplete data.
2. How can random sampling be used to expedite the running time at a small cost on accuracy, and how large should the sample size be?
3. How can shortest-path metric be used to avoid computing distances between all pairs? How will one construct the underlying graph? How well does it work?
4. When the input trajectories have high complexity, computing the distance can be time-consuming. How can the trajectories be simplified into fewer points while controlling the amount of error incurred by such a transformation?
5. Linear interpolation does not work well for traffic in urban environments, especially when data is noisy or incomplete (i.e., many of the sample points are missing). Suggest interpolation schemes that exploit the fact that we have a large set of trajectories and not just one trajectory.

6.1 What to do (Code)

1. To keep the previous code for Parts 1-3 intact, first copy all of the source files into a new directory dedicated for Part 4.
2. Implement the proposed changes and experiments to measure the impact of the changes.

6.2 What to do (Slides)

1. Clearly state the aim of each of the two extensions.
2. Propose modifications and state the hypotheses for their impact.
3. Compare and contrast the modified algorithm with the baseline algorithm (Parts 1-3) by conducting experiments. You may use our provided datasets or generate your own; see Section 10. Include empirical data and analysis in the slides. Describe the strengths and weaknesses of the modification.
4. Discuss whether the hypotheses held. Why or why not?
5. Describe potential avenues for further work towards the stated aims.

7 Requirements

- ***Groups***: The project is to be done in groups of ***four***, ideally composed of two homework pairs.
- ***Programming languages***: We recommend that all groups use Python, but you may use C/C++, R, Matlab, or Java. Please contact Alex (asteiger@cs.duke.edu) if your group would like to use a different language than one listed here!
- ***Submission***: Exactly ***ONE*** group member will upload ***two*** files to Sakai:
  1. Your presentation slides in `.ppt` or `.pptx` (PowerPoint) format that includes all experimental analyses, discussions, figures, and "speaker's notes" per slide. The majority of the text should be in the speaker's notes. Google Slides, OpenOffice, LibreOffice, and Keynote can export to the required formats.
  2. A single ZIP archive that includes all of your source code and a README file that describes all compilation instructions, execution instructions, and the organization of your code (e.g. what each file contains). If you use any datasets that we do not provide, include links to them if they are accessible online, or include the dataset with your submission if you generated them.

8 Evaluation

This project is worth 15% of your overall course grade. The weight of each Part is as follows:

- 20%: Part 1 – Computing distance
- 20%: Part 2 – Center Trajectories
- 20%: Part 3 – Clustering
- 30%: Part 4 – Extensions
- 10%: Organization and Presentation

Parts 1-3 will be evaluated on the correctness of the implementations of the given procedures, the interpretability of visualizations/figures, and quality the experimental analyses.

Part 4 will be evaluated on the clarity of the stated aims, the soundness of the modifications towards the aims, and the detail and effort of the experimental analyses and conclusions.

The final category will be evaluated on the organization of the ZIP archive and the presentation of experimental data, analyses, and discussions in the slides, including text written in the "speaker's notes."

9 Getting Started

Download the `highway.csv` dataset from Sakai. This is a synthetic dataset modeling traffic on an 8-lane highway. The file is in CSV format, where the first line describes the names of the four columns,

and each other line encodes one point of a trajectory and the lane number corresponding to the trajectory. For example, suppose trajectory T with id 7 from lane 3 is represented by $\langle(0,1),(3,4),(8,9)\rangle$. In the format used by `highway.csv`, T is represented by the rows:

```
7,0,1,3
7,3,4,3
7,8,9,3
```

This dataset is well-behaved and suitable for the first testing of your implementations. ***NOTE***: The lane numbers (cluster IDs) are NOT to be used within your procedures; they are to be used strictly for testing and analysis.

Parsing CSV files is a common task is likely supported with standard libraries (e.g. Python) or by popular third-party libraries (e.g. OpenCSV for Java).

10 Datasets

Here is a list of potentially useful datasets for you to test your implementations and perform experiments on. Please share any other datasets you find interesting on Ed!

- `highway.csv` on Sakai is the I5_SIM dataset by Morris et al., 2009. The trajectories are synthetic and model traffic on an 8-lane highway. Converted from Matlab format.
  Source: http://cvrr.ucsd.edu/bmorris/datasets/dataset_trajectory_clustering.html
- `energy.csv` on Sakai is a small, cleaned subset of the Vehicle Energy Dataset by Oh et al., 2020. The trajectories are recorded from GPS trackers in vehicles on a road network. Converted from Excel format.
  Source: https://github.com/gsoh/VED
- GeoLife GPS Trajectories by Microsoft Research Asia, 2012. The trajectories are recorded from individuals' GPS trackers in a broad range of outdoor activities.
  Source: https://www.microsoft.com/en-us/download/details.aspx?id=52367