Trajectory Clustering and Insight Visualization

Quan Hoang Ngoc - 22521178

October 2024

Keywords: *Trajectory Clustering, Data Analysis, Pattern Recognition, Movement Patterns, Partition-and-Clustering*

1 Introduction

Recent advancements in satellite technology and tracking systems have led to an exponential increase in the availability of trajectory data from various sources, including vehicle movements, hurricane paths, and animal migrations. As this data grows in volume and complexity, analyzing it becomes essential for uncovering meaningful insights. Identifying common movement patterns is crucial for diverse applications, such as autonomous vehicle navigation, urban traffic management, natural weather pattern discovery, and animal behavior analysis.

Trajectory clustering has emerged as an effective approach to extract insights from such data. It provides visual insights, which are often more intuitive and accessible than statistical results, allowing both experts and non-experts to explore and analyze movement patterns.

However, standard clustering methods encounter significant challenges as the variety and volume of trajectory data continue to grow, sourced from GPS systems, IoT devices, and other technologies. Trajectories are often complex, covering long distances with irregular directions, making it difficult to uncover meaningful patterns by treating the whole trajectory as a single data point. A key limitation of standard clustering algorithms is their inability to capture partial patterns within a trajectory. Note that common patterns can be a part of a trajectory while the whole trajectory is not. Thus, using the entire trajectory as a data point may miss important patterns, leading to a loss of valuable insights.

These observations motivate our approach: partition and clustering. We partition individual trajectories into line segments and then use these segments to cluster and find groups of similar movement. This means that each trajectory can belong to multiple clusters, and the grouped segments can represent common patterns. This method is especially advantageous for discovering frequent sub-trajectories within a dataset, offering clearer insights than raw trajectory data.

In summary, our contributions to the field of trajectory data analysis include:

- **Partition-and-Clustering Framework:** We propose a novel trajectory clustering algorithm that partitions each trajectory into smaller line segments and clusters these segments to identify patterns. This approach effectively handles complex trajectories with multiple directions, long paths, and large datasets.
- **Discovering Frequent Patterns:** Our method focuses on sub-trajectories to reveal common movement paths, offering a new perspective for exploring associations and patterns beyond traditional statistical methods.
- **Effective Trajectory Partitioning:** We demonstrate a method for partitioning trajectories into meaningful segments, serving as an example of data normalization and reduction.
- **Custom Distance Function and Indexing:** We design a specialized distance function for clustering line segments and propose an efficient indexing method that calculates the density of segments by treating them as geometric objects.

- **Visual Representation of Patterns:** After clustering, we represent common patterns as visually interpretable movement paths, making the results accessible for applications such as traffic analysis, path optimization, and behavior modeling.

2 Methodology

The problem of trajectory analysis through partitioning and clustering can be broken down into two independent yet essential tasks: **Individual Trajectory Partitioning** and **Line Segment Clustering**. While clustering can leverage established algorithms by mapping the task to previously solved problems (e.g., DBSCAN), trajectory partitioning presents a unique research challenge. Effectively partitioning trajectories requires deeper attention since it directly impacts the quality of clustering and the insights gained from the data.

2.1 Trajectory Partitioning: A Critical Step

A trajectory is fundamentally a sequence of points representing the movement of an object over time. Partitioning involves identifying specific *characteristic points* along this sequence to divide the trajectory into smaller, meaningful segments. Instead of retaining every point in the original sequence, only these characteristic points are used to reconstruct the trajectory as a path of line segments. This approach provides several key benefits:

- **Data Normalization and Reduction:** By focusing only on characteristic points, the trajectory becomes more concise, eliminating unnecessary data points and simplifying the analysis.
- **Enhanced Interpretability:** Replacing complex, noisy trajectories with clean line segments helps highlight significant movement patterns, making the data easier to interpret.
- **Improved Processing Efficiency:** With fewer data points, the computational overhead is reduced, enabling faster clustering and analysis.

This partitioning process transforms raw trajectory data into a simplified form that retains the essential movement characteristics, paving the way for more efficient clustering and insightful visualization.

Challenges in Selecting Characteristic Points

Selecting the right characteristic points is crucial, as they must accurately represent the trajectory's key movements while minimizing information loss. This trade-off between data reduction and integrity is a central research challenge in trajectory partitioning. An effective trajectory partitioning method must meet two essential criteria:

- **Minimal Number of Characteristic Points:** The partitioning should select as few characteristic points as possible to reduce the complexity of the trajectory.
- **Minimal Error Between Original and Reduced Paths:** The reconstructed path of line segments, formed by connecting characteristic points, should closely approximate the original trajectory. This ensures that important movement patterns are preserved.

To balance these two requirements, we define a *loss function* that captures both the length and accuracy of the reconstructed path. Specifically, the loss function considers the total length of the line segments (L2 norm) and the total error between the original points and the replaced line segments.

Quantifying the Error in Trajectory Partitioning

To accurately quantify the error between a line segment and the points it replaces, we need to consider two components:

1. **Distance Error:** A simple way to measure error is by calculating the total distance between the original points and their corresponding line segment. This reflects how closely the line segment approximates the sub-trajectory. However, using only distance may not fully capture the *directional difference* between the original movement and the reconstructed segment. Additionally, we compute the **F1-mean** for the distance of two adjacent points instead of individual points.

2. **Direction Error:** To account for directional differences, we introduce a *penalty for angular error* between the line segment and the sub-trajectory. For this, we compute the **sine of the angle** between the line segment and each sub-line of the sub-trajectory (i.e., segments formed by adjacent original points).

    - Scaling by Sub-Line Length: Since longer sub-lines contribute more to the overall trajectory structure, we scale the angular error by the length (L2 norm) of the sub-line. This ensures that larger directional deviations in longer sub-lines are penalized more heavily.

This multi-faceted error quantification approach allows for a more accurate representation of the trajectory while balancing data reduction and the integrity of significant movement patterns. To further refine this balance, we introduce weighting factors, α_1 and α_2, to scale the contribution of each error type, rather than using equal weights. Specifically, the total error can be expressed as:

$$\text{Error} = \alpha_1 \cdot \text{Distance Error} + \alpha_2 \cdot \text{Direction Error} \tag{1}$$

Here, α_1 and α_2 are adjustable hyperparameters that can be tuned based on the relative importance of distance accuracy versus directional accuracy for a particular application. By setting appropriate values for α_1 and α_2, we can prioritize either minimizing spatial deviations (distance error) or maintaining the original movement's directionality (direction error).

This flexibility in error weighting ensures that our partitioning approach remains adaptable across various use cases, where different aspects of the trajectory data may hold varying degrees of significance. Consequently, the proposed method provides a more tailored trajectory representation that accommodates diverse requirements in trajectory analysis.

2.2 Greedy Approximation

The task of finding a set of characteristic points that minimize the loss function is inherently challenging, as it is classified as an *NP-hard* problem. To address this, we adopt a *greedy strategy* to find an approximate solution that focuses on local optimization rather than achieving a true global optimum.

An important observation is that the *starting point* of a trajectory is always a characteristic point. With this in mind, we will iteratively determine the next characteristic point by optimizing locally from the previous characteristic point. Following the same approach, the subsequent characteristic points are found based on their preceding characteristic point, continuing this process throughout the trajectory.

Applying the greedy principle, we evaluate whether replacing the sequence of points from the initial point to the j^{th} point with a line segment formed by the initial point and the j^{th} point results in a better approximation. If this replacement satisfies the criteria for improvement, then we choose to make the replacement. The criteria for selecting the next characteristic point, denoted as j, are as follows:

- The error associated with replacing the sequence of points from the initial point to the j^{th} point with the line segment connecting these two points (including the length of this line segment) must be `less equal than` the total length of the sub-trajectory created by these points. This ensures that the approximation is beneficial.
- To satisfy the Minimal Number of Characteristic Points criterion, we select the `largest possible value of` j for which the replacement remains effective. In other words, the second characteristic point is chosen as the furthest j^{th} point such that the line segment (formed by the initial point and the j^{th} point) accurately approximates the sub-trajectory with an error (including the length of itself) smaller than the length of the original sub-trajectory.

This iterative process continues until the entire trajectory is partitioned, ensuring that each new characteristic point is selected in a manner that optimizes the local approximation while aiming to minimize the overall number of characteristic points used. By following this approach, we achieve an effective balance between trajectory simplification and data integrity.

2.3 Line Segment Clustering

Given the nature of trajectory data, using a density-based clustering algorithm to group segments with similar movement patterns is a natural choice. In this study, we utilize an instance of the **DBSCAN** algorithm to identify clusters of line segments. To effectively apply DBSCAN, several considerations are necessary:

- **Density Definition:** Unlike points in traditional feature spaces, line segments represent directional movements defined by two endpoints. Thus, they should be treated as geometric objects rather than individual points in a feature space.
- **Hyperparameter Tuning:** Properly setting hyperparameters such as the DENSITY RADIUS and MIN SAMPLES is crucial for achieving optimal clustering results. These parameters determine the minimum neighborhood size and density requirements for forming clusters.

Designing the Distance Function

To measure the similarity between two line segments, we extend the error function previously discussed, adapting it to be suitable for comparing two segments that play equal roles. The original error function is primarily designed for assessing the fit between a sub-line (usually shorter) and a longer line segment. Therefore, additional adjustments are needed for a fair comparison between two segments of similar status.

The proposed distance function incorporates the following components:

- **Length Error:** We introduce a penalty based on the difference in lengths between the two line segments. This is defined as the minimum length difference when projecting one segment onto the other. By accounting for length disparities, we ensure that longer deviations are more heavily penalized.
- **Symmetry Consideration:** To satisfy the permutation property of a distance function, we calculate the total error for both possible roles, i.e., treating the first segment as the primary and then swapping their roles. The final distance is taken as the average error from these two configurations.

This customized distance metric provides a robust foundation for clustering line segments based on their geometric properties, capturing both spatial and directional characteristics. It ensures that the DBSCAN algorithm can accurately identify groups of segments that exhibit common movement patterns.

Efficient Indexing

The **DBSCAN** algorithm requires efficient indexing of line segments to accurately and quickly retrieve the density for each segment. Traditional indexing structures, such as the **KD-Tree**, are optimized for Euclidean distance; however, our customized distance metric poses a unique challenge for direct application in these structures.

To address this issue, we propose a **Two-Phase Retrieval** approach for efficient indexing:

1. **Feature Point Transformation:** Each line segment is reformatted as a feature point in a high-dimensional space. This transformation enables the use of traditional indexing techniques.
2. **Approximate Neighbor Retrieval:** We employ the **KD-Tree** structure with Euclidean distance to quickly identify the top-k nearest neighbors for each segment. Although the retrieved neighbors are based on the Euclidean distance metric, this step serves as an efficient pre-filtering process.
3. **Custom Distance Recalculation:** Once the top-k neighbors are obtained, we recalculate the distance using our customized distance function to refine the results. This step ensures that the final nearest neighbors align with our designed metric, preserving the accuracy of the clustering process.

The Two-Phase Retrieval approach allows for efficient indexing while accommodating the complexities introduced by the custom distance metric. By leveraging the fast retrieval capabilities of **KD-Tree** and then refining our novel distance measure, we achieve a balance between computational efficiency and clustering accuracy.

2.4 Visual Patterns Representation

Finally, as previously discussed, after clustering line segments into groups, we aim to identify groups containing multiple trajectories, which are likely to represent common movement patterns. To convey these patterns effectively, we need a visualization method that is both intuitive and user-friendly, providing valuable visual insights. We will represent each group's line segments as a single path constructed from *representation points*, offering a visual summary of the common movement pattern.

To achieve this, we follow a systematic process based on the **average direction vector** of the line segments within the group, which serves as the general direction for the entire cluster. The steps for transforming the original points using this vector are as follows:

1. **Projection onto the Average Direction:** Each point from the line segments is projected onto the computed average direction vector. This projection aligns the points along a new coordinate axis defined by this vector, enabling a uniform analysis of the line segments' structure.
2. **Calculation of Average Height:** For each projected coordinate in the new coordinate axis (denoted as x'), we calculate the average value of the corresponding orthogonal coordinate (y'). This averaging procedure generates a set of *representation points* that succinctly summarize the group's movement characteristics.
3. **Reconstruction in the Original Coordinate System:** The representation points are then transformed back to the original coordinate system by rotating them from the new coordinates (x', y'). The final step involves sequentially connecting these transformed points, resulting in a simplified path that accurately represents the original group's movement pattern.

This approach effectively generates a visually intuitive representation by aligning the original line segments with the primary direction of movement. The resulting path serves as the *common movement pattern* for the group, capturing essential characteristics while filtering out noise and irregularities.

3 Experiment

4 References