2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN) | 978-1-6654-9624-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/IPSN54338.2022.00043

Clustering of Trajectories using Non-Parametric Conformal DBSCAN Algorithm

Haotian Wang
Rutgers University
hw487@rutgers.edu

Jie Gao
Rutgers University
jg1555@rutgers.edu

Min-ge Xie
Rutgers University
mxie@stat.rutgers.edu

ABSTRACT

Technology innovation has provided the opportunity to study the characteristics of natural human mobility. In this paper, we look at how to identify interesting clusters (by different individuals or other naturally defined groups) in a family of trajectory traces. We focus on coarse-grained, sparsely sampled trajectories inferred from sporadic occurrences in an unsupervised setting. This is a challenging setting due to difficulties in selecting features and similarity measures, and due to lack of prior knowledge of data distribution. We propose a non-parametric clustering algorithm, which makes little assumptions on prior knowledge of both data distribution and cluster properties. Our algorithm, *Conformal DBSCAN*, combines density-based DBSCAN clustering with the statistical conformal prediction framework. We first identify groups of highly similar trajectories as the initial seeds of clusters, similar to DBSCAN. Then we include additional trajectories that belong to this cluster, with a guaranteed statistical confidence level, derived by an improved conformal prediction framework. This allows the clustering algorithm to automatically adapt to different data distributions. Our algorithms are shown to significantly outperform alternative clustering algorithms on several artificial and real-world datasets.

1 INTRODUCTION

Technology innovation has enabled the possibility to collect a large number of human trajectories. These trajectory traces reveal interesting characteristics of natural human mobility that are of significance both on an individual level (personal mobility profiling) and on a population level (for gathering and group motion) with numerous applications such as traffic engineering [35, 40], civil planning [23] and public health domains [25]. Trajectory data has been collected by vastly different sensing modalities and has a high variation of spatial and temporal resolution. With dedicated sensors either carried by the agents (e.g., GPS) or carefully instrumented in the environment (e.g., in smart buildings), one can obtain high-resolution spatial-temporal trajectories [11, 29]. On the other hand, a larger category of trajectory traces are inferred from sporadic, discrete occurrences with or without timestamps or orderings – for example, trajectories inferred from proximity with WiFi access points, cellular towers, highway toll stations or public transit ticket booths [26, 28, 42, 48]. This kind of trajectory data is sparse, not uniformly sampled or even disconnected, and lacks details.

Many methods on analysis of trajectory data focus on high-quality trajectory data. There are a lot of rich details and geometric features one can extract, especially with supervised learning. Past work has studied the prediction of the transportation modes [49, 50], turning mode in the crossroad [27], and destination of the trips [5], These supervised learning methods rely on detailed discriminative statistical features such as velocity [49] and acceleration [19, 50], which can only be extracted from fine-grained trajectories with very dense sampling intervals.

The work in this paper focuses on coarse-grained, sparsely sampled trajectories, or sparse samples of occurrences. We also take an unsupervised setting without any training data or labels. We ask whether one can still infer meaningful clusters in a given set of trajectory traces. A cluster can correspond to trajectories by the same individual or other naturally defined groups (e.g., the same animal species). On the scientific front, we believe that different individuals or species move in a different way and there are hidden features in the traces that can be used to separate them.

1.1 Challenges

The immediate challenges in handling sparsely sampled trajectories are to decide on feature selection, distance measurement policies, and clustering algorithms.

Feature selection. The first challenge is to find a good representation of an input trajectory. Existing techniques mainly use spatial and temporal features of the trajectory [17, 22]. These low-level position based features are sometimes insufficient to distinguish differences in a small region [46]. Semantic information is taken into consideration, such as the categories of check-in locations [6, 44] and transportation mode parameters [49, 50]. But advanced semantic information requires external labels which are not always available. Advanced deep learning techniques can be used to train distinguishing feature vectors [43, 47], but falls short in transparency and interpretability. In general, feature design for clustering trajectories is still a non-trivial problem. Although we do expect that mobility trajectories of the same individual tend to be repetitive and regular, there are clearly daily variations and outlier behaviors. It is unclear what are the most discriminating features that separate the mobility patterns of different individuals.

Trajectory similarity. To compare two trajectories, classical geometric measures for distances of curves use geometric definitions such as the Hausdorff distance and Fréchet distance [38]. These measures are more suitable on trajectories that are aligned, uniformly and consistently sampled, such as trajectories derived from video footage [3, 24, 46]. For sparsely sampled trajectories with potential missing data and outliers, these measures face challenges. Specifically, these distance measures are *extreme* measures (of the min-max or max-min type) and capture the worst-case scenario. On a real-world human trajectory dataset, it is nearly impossible for two trajectories to be close to each other all the time. These geometric measures often turn out to be too large (e.g, comparable to the radius of the city) to be interesting [13]. Even for two trajectories of the same individual on two consecutive days, the Hausdorff distance and Fréchet distance are on average in the order of several kilometers [42] and thus can not be effectively used to differentiate

978-1-6654-9624-7/22/$31.00 ©2022 IEEE
DOI 10.1109/IPSN54338.2022.00043

trajectories from different agents. Motivated by this, several recent works [33, 39, 42] define similarity measures for real-world trajectories, by relaxing the dependency on extreme conditions (e.g., by considering partial similarity measures).

Clustering algorithm. Existing clustering algorithms are not particularly suitable for trajectory data. Most clustering algorithms in the literature fall into the following two categories. The first one is model-based. One of classical methods is centroid-based methods. The clusters are inherently assumed to be round and the problem is often formulated as an optimization problem to minimize the maximum size of each cluster, such as k-center [15] or k-means clustering [31]. These algorithms generate less meaningful results when the ground truth has other 'shapes' or of different sizes [12]. Another model-based approach assumes the distribution of data points inside each cluster [9], e.g., in the mixed Gaussian model. The clustering algorithm explicitly uses this assumption on input data, which is not applicable for trajectory data.

The other class is density-based methods, e.g., density-based spatial clustering of application with noise (DBSCAN) [10]. The idea is to first identify the core points where the number of points within distance r from each core point is more than a given threshold. Then the cluster grows by including all points that are within distance r from at least one core point in the cluster. The clustering results are influenced by parameter selection and these parameters need to be determined beforehand [2]. A few variations of DBSCAN choose the input parameters automatically. For example, a hybrid DBSCAN algorithm incorporates Binary Differential Evolution and Gaussian Means to determine the parameters [36]. Additional research works discuss this automatic selection process, using grid partition technique [18], affinity propagation clustering [7], domain sets [16] and other methods [20, 37, 51]. These methods mainly help to choose appropriate fixed parameters for clusters before the growing process. The parameters are not changed during the growing process.

We propose our method, which follows the general framework of DBSCAN, but adapts the parameter r automatically in the growing process. We grow a cluster C by including the points that are, statistically, believed to be taken from the same distribution that generated C with high confidence. This allows the clustering algorithm to adapt to different cluster distributions and leads to improved clustering performance (measured by the Adjusted Rand Index) on different trajectory datasets. We highlight our contributions below.

1.2 Our Contribution

Our goal is to take a non-parametric approach – as we do not have any prior knowledge of the ground truth clusters of trajectory traces (e.g., cluster labels, the number of clusters, the distribution of clusters) – and support a wide range of features and similarity measures on real-world trajectory datasets. Our cluster shall move beyond a centroid-based definition and shall tolerate variation in data density, the existence of outliers, and variations in distributions.

We first take a look at Figure 1 to understand where current algorithms fall short. The figure shows three sample datasets and four algorithms: k-means algorithm, Gaussian mixture models (GMM), DBSCAN and our *conformal DBSCAN* algorithm. k-means fails to identify high-density clusters that are not of a round shape and have varying densities. GMM successfully finds the three Gaussian clusters but shares similar limitations with k-means for other non-round clusters. DBSCAN, on the other hand, can successfully recognize 'gaps' between clusters. But the algorithm is local – a new data point x will join an existing cluster C if x is considered to be similar (within distance r for some distance measures) to an existing core data point x' in C, while r is a fixed value. Thus, low-density points (in the second and third columns) are either lumped together with the high-density points (if r is too big) or not recognized by any cluster at all (if r is too small).

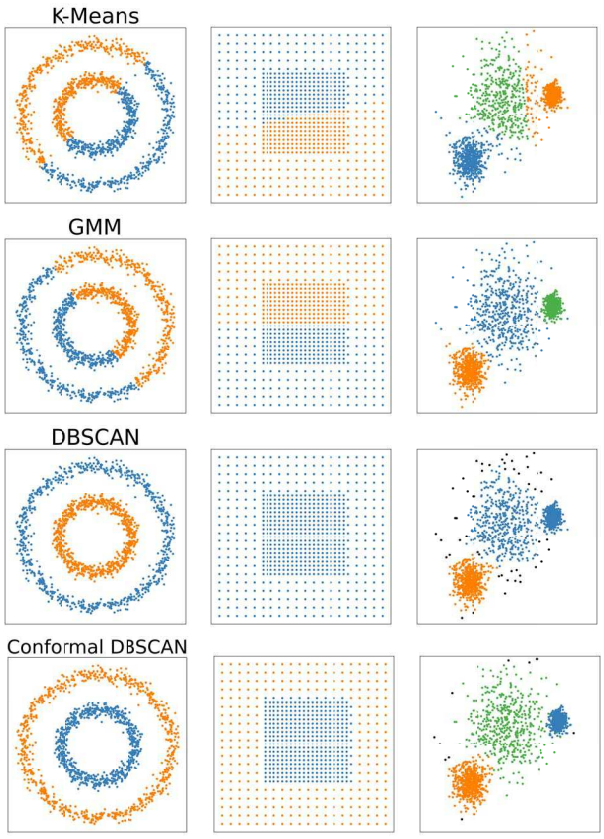


Figure 1: Four algorithms: k-means, Mixture of Gaussian (GMM), DBSCAN and conformal DBSCAN on three synthetic datasets. The clusters are shown in different colors.

Then, let's discuss the challenge in trajectory clustering. For trajectory data, there is no standard definition of clusters and we do not have any prior knowledge of the distribution of trajectories that belong to the same individual. For supervised learning, one can use an empirical distribution of the history data, but this is unavailable for the unsupervised setting. Here we assume that there are k clusters $C_1, C_2, \cdots, C_k$, with k unknown, and the points in C_i are uniformly randomly sampled from a distribution π_i, which is also unknown. Notice that the distributions π_i for different i

are sufficiently separated; otherwise, they shall have already been merged to fewer clusters.

We take the DBSCAN algorithm but improve the growing step significantly – starting from an initial set of high-density points, we include new ones with an adaptive radius r, which is determined by the current cluster properties. This is achieved by using the conformal prediction framework, a statistical framework to quantify the likelihood of a new data point belonging to a distribution π by comparing with random samples independently and identically taken from π. Thus our algorithm is called *conformal DBSCAN* algorithm.

The conformal prediction framework [41] is a recent development in statistics and machine learning that quantifies the likelihood of a new data object x belonging to an unknown distribution $\mathcal{D}$, where the only information needed is a set C of data objects randomly sampled from the distribution. The framework does not need any assumption on the knowledge of the distribution $\mathcal{D}$. The conformal prediction framework uses a *discrepancy score* $\mathcal{A}(x, Y)$ which characterizes how different a data object x is relative to a reference set Y. If two data objects are similar, they will have similar discrepancy scores with respect to the same reference set. Thus, we compare the new data object x with each object $c \in C$ (with respect to the rest of objects in C as the reference set) – if x is also likely randomly chosen from $\mathcal{D}$, roughly a similar amount of the objects in C are expected to have discrepancy scores higher or lower than x. The rank of its discrepancy score among all the objects in C provides a rigorous framework to quantify the likelihood of x being an element from $\mathcal{D}$, which also controls the type-I error of labeling. Notice that this allows a wide range of designs for the discrepancy score and covers a wide range of (unknown) distribution $\mathcal{D}$.

To handle the growing phase in DBSCAN we need to modify the conformal prediction framework. Here the initial cluster includes a set of high-density points, instead of randomly selected from a cluster, which is required in the conformal prediction framework. We provide confidence analysis for this more general and distorted setting. We reformulate the conformal prediction test and include a new point x if it is within distance r from a point $c \in C$ whose discrepancy score *ranks* within $[\varepsilon|C|, (1-\varepsilon)|C|]$ among all points in C, where ϵ is a small number and is interpreted as the significance level in our work. We prove that a newly included object belongs to the current cluster with a high significance level ϵ. In other words, the type-I error of this labeling, false positive rate, is controlled below ϵ in the growing process. Our modified conformal prediction framework is applied in an iterative manner to group objects into several clusters. Our work is the first one to combine the conformal prediction framework and unsupervised clustering method together, with theoretical analysis for the growing step in the DBSCAN method.

We also tested our algorithm for different trajectory datasets. For identifying meaningful clusters, we use three different trajectory similarity measures that are robust to missing data and outliers . They can capture different innate features (spatial and temporal features) of mobility trajectories. Three trajectory datasets, including synthetic trajectories and real-world trajectories (wild animal trajectories and individual electric bike trajectories), are tested. Through the experiments, our method significantly outperforms other baseline algorithms (k-means and DBSCAN [10]). In the animal species trajectory dataset, our method can separate the trajectories by their species, with Adjusted Rand Index (ARI) value as 0.7449, much higher than the other methods. In the bike trajectory dataset, our goal is to group each individual's daily trajectories without any prior label. Our method performs well using different similarity measures.

In the rest of this paper, we start by reviewing the previous work, DBSCAN and the conformal prediction framework, in Section 2. In Section 3, we proposed our conformal DBSCAN algorithm with theoretical analysis. The relationship with DBSCAN is also discussed. The experiments with different trajectory datasets are presented in Section 4 and Section 5 concludes this paper.

2 REVIEW OF DBSCAN AND CONFORMAL PREDICTION

In this section, we will review DBSCAN and the conformal prediction framework. We also discuss their limitations for clustering.

2.1 DBSCAN

DBSCAN is a heuristic algorithm widely used in data mining and clustering. The main idea of DBSCAN is the following: given a collection of data objects and two predetermined fixed parameters, the radius of neighborhood r (in a distance measure) and the threshold for the number of neighbors $minPts$, the points are gradually placed into clusters. To find one cluster C we take the following steps:

(1) *Find the core set:* First, if x has at least $minPts$ points within radius r (including x itself), x and all points within distance r from x are put in C. x is called a core point.
(2) *Gradually include reachable points:* A point x not yet included in any clusters is added to C if x has at least one core point of C within distance r.

The above procedure finds one cluster C in DBSCAN. When C cannot grow anymore, we find another high-density region and repeat. The procedure stops when no more clusters can be identified. The points that are not recognized by any clusters are considered as noise or outlier.

The two parameters in DBSCAN are fixed for all clusters. Later variants to DBSCAN choose different parameters, i.e., the radius r and the threshold $minPts$, for different clusters but the parameters are fixed throughout the growing phase for one cluster. In our algorithm, we use an adaptive parameter r in the growing phase. The choice of r is guided by the points in the current cluster C. This is determined by using a revised conformal prediction framework, which is introduced below.

2.2 Conformal Prediction

The conformal prediction framework is a statistical test to determine whether a new object belongs to a distribution with *i.i.d.* observation objects. Our discussion focuses on the *Jackknife+ conformal prediction framework* [4], introduced below.

Let $C = \{c_1, c_2, \cdots, c_m\}$ be a given set of objects from a single cluster with underlying distribution $\mathcal{D}$. which is unknown. We would like to check whether or not a new object $c_{m+1} \notin C$ is also a uniform random sample from the same distribution $\mathcal{D}$.

First, we define the notion of *discrepancy score* computed by an algorithm or a function $\mathcal{A}(x, Y)$, which characterizes the data object x with respect to a reference dataset Y, $x \notin Y$. Notice that there is no assumption on this discrepancy score in this framework and our following clustering algorithm. It can be any meaningful measure that characterizes the distance of x with Y, the role of x within set Y, the influence of x upon Y, etc. This discrepancy score can be derived from certain distance measures using extracted features, or as an application specific score (e.g., as the output of a preprocessing algorithm such as regression or neural network models). While the conformal prediction framework works with *any* real-value function $\mathcal{A}(x, Y)$ as the discrepancy score, when an appropriate discrepancy score is chosen, the prediction could be more informative. There are some popular choices in the literature, such as the k-nearest neighbors algorithm [41], (kernel) ridge regression [41], SVM [41], neural networks [30], random forest [8] and genetic algorithms [21].

Next, we compare the discrepancy score of all the objects to quantify the probability that a new object belongs to the same distribution as the reference set. If the new object c_{m+1} belongs to the cluster C, we can compare the discrepancy score $\alpha_{m+1}^{(i)}$ and α_i, where $\alpha_{m+1}^{(i)} = \mathcal{A}(c_{m+1}, C^{-i})$ and $\alpha_i = \mathcal{A}(c_i, C^{-i})$ with respect to the same leave-one-out set $C^{-i} = C \backslash \{c_i\}$, and $c_i \in C$ is a randomly selected object from the cluster C. If both c_{m+1} and c_i are random samples from the same distribution, the order of $\alpha_{m+1}^{(i)}$ and α_i would take a 50-50 chance. In other words, when the new object c_{m+1} is similar to the objects in the cluster C, we expect the value of $\alpha_{m+1}^{(i)}$ is comparable to a sizeable fraction of the value of α_i, for all $i = 1, 2, \cdots, n$. In the cases when $\alpha_{m+1}^{(i)}$ is larger or smaller than the majority of α_i values, we would reject the claim that c_{m+1} is the same or similar to the objects in C.

Formally, we count the numbers of values α_i for all i, that $\alpha_{m+1}^{(i)} < \alpha_i$ and $\alpha_{m+1}^{(i)} > \alpha_i$, respectively. Then we define a *conformity score* in the range of $[0, 1]$ as

$$p(c_{m+1}) = 2 \min\{q^-(c_{m+1}), q^+(c_{m+1})\} \tag{1}$$

where

$$\begin{aligned} q^-(c_{m+1}) &= \frac{\sum_{i=1}^m \mathbf{1}\{\alpha_{m+1}^{(i)} < \alpha_i\}}{m} + \frac{\sum_{i=1}^m \mathbf{1}\{\alpha_{m+1}^{(i)} = \alpha_i\}}{2m} \\ q^+(c_{m+1}) &= \frac{\sum_{i=1}^m \mathbf{1}\{\alpha_{m+1}^{(i)} > \alpha_i\}}{m} + \frac{\sum_{i=1}^m \mathbf{1}\{\alpha_{m+1}^{(i)} = \alpha_i\}}{2m} \end{aligned} \tag{2}$$

and $\mathbf{1}(\cdot)$ is the indicator function. When c_{m+1} is similar to c_i, we expect that $\alpha_{m+1}^{(i)} \leq \alpha_i$ or $\alpha_{m+1}^{(i)} \geq \alpha_i$ holds by roughly the 50-50 chance. So, if either $q^-(c_{m+1})$ or $q^+(c_{m+1})$ is close 0, it means c_{m+1} is likely different than majority objects in C. In either of the two cases, the conformity score $p(c_{m+1})$ is small and close to 0.

An example is provided in Figure 2, where C contains points on a circle. When new points are off the circle (e.g., c_9, c_{10}), both the conformity scores $p(c_9)$ and $p(c_{10}) \approx 0$. For new points on the circle (e.g., c_{11}), the conformity score $p(c_{11}) \gtrapprox 0$.

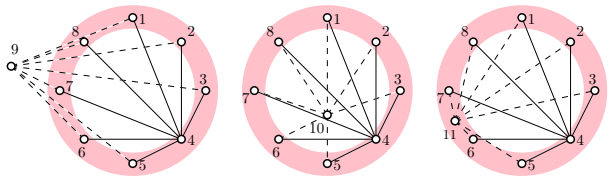


Figure 2: Illustration of the conformity score. There are 8 points, $c_1, \ldots, c_8$ in the reference set C, with the underlying distribution on a circle (shown in pink). The points c_9 and c_{10} are checked against a randomly chosen point (say c_4). $\mathcal{A}(x, Y)$ is the sum of distances from x to all points in Y. $\alpha_9^{(4)} > \alpha_4$; $\alpha_{10}^{(4)} < \alpha_4$ and $\alpha_{11}^{(4)} \approx \alpha_4$. In fact, both $p(c_9)$ and $p(c_{10})$ are close to zero; while $p(c_{11})$ are away from 0.

In the case when $c_1, \ldots, c_m$, the points in C, as well as the new object c_{m+1} are all independent random draws from the same distribution, it can be shown [4, 14] that

$$\Pr(p(c_{m+1}) \leq \epsilon) \leq \epsilon, \tag{3}$$

where ϵ is a pre-specified small number in $(0, 1)$. That is, if we use the detection rule to declare "c_{m+1} is a sample point from the same distribution as those in C" if and only if "$p(c_{m+1}) > \epsilon$", then the type-I error to mistakenly reject c_{m+1} is less than ϵ. In fact, some researchers suggested to interpret the conformity score $p(c_{m+1})$ as a p-value for the hypothesis test problem H_0: c_{m+1} *is conformal with* C versus H_a: c_{m+1} *is not conformal with* C [14, 45].

The above guarantee often requires that the points in C and new point c_{m+1} are drawn from $\mathcal{D}$ at random. Intuitively, for a randomly drawn x from $\mathcal{D}$, the discrepancy score $\mathcal{A}(x, C)$ forms a distribution π in $\mathbb{R}^1$. Regardless of what π or $\mathcal{D}$ look like, randomly drawing m samples from $\mathcal{D}$ will give us a random sample of discrepancy scores from π. Thus, we can use the *ranking* of $\mathcal{A}(c_{m+1}, C)$ with a random sample from π to estimate the likelihood that c_{m+1} is also taken from the same distribution $\mathcal{D}$. Notice that using the ranking rather than the discrepancy score itself allows for a wide variety of discrepancy functions and embraces inherent robustness. The Jackknife+ framework made a slight adjustment with C replaced by C^{-i} when we compare c_{m+1} against an object c_i from C, to avoid the influence of $c_i \in C$ in the discrepancy calculation, which ensures a rigorous mathematical proof of the statistical claims.

However, for the clustering problem, we have no prior knowledge of the distributions and the cluster labels. Thus we do not have a reference set to start with. Generally, we can guess the label of random samples from a truncated distribution – with density function higher than a threshold – and use them as the reference set. This is similar to the first step of DBSCAN of discovering the high-density regions. The issue is that the samples identified from the truncated distribution are not a uniformly random sample from the ground truth cluster. This violates the assumption of conformal prediction. For a new object on the boundary of this truncated distribution, its discrepancy score is likely to be far away from the majority of the discrepancy score of objects in the reference set. In the following work, we will extend the conformal prediction framework and provide the mathematical explanation for the newly added objects with respect to a truncated distribution.

Algorithm 1: Conformal DBSCAN

Input: The set of data objects $S = \{c_1, \ldots, c_n\}$;
A small number $\epsilon \in (0, 1)$;
Any radius value δ to select the initial seeds;
Output: The set of clusters $\{C_1, C_2, \ldots\}$

1 $k = 1$; // Index of the current cluster
2 **while** *S is not empty set* **do**
// Initial members
3 $N(i) \leftarrow \{c_j | d(c_i, c_j) \leq \delta\}$, for $c_i \in S$;
4 $t \leftarrow argmax_i |N(i)|$;
5 $C_k \leftarrow N(t)$;
6 **repeat**
// Core objects determination
7 $\alpha_i^{(j)} = \mathcal{A}(c_i, C_k \setminus \{c_i, c_j\})$, for $c_i, c_j \in C_k$;
8 $p(c_i) = \frac{\sum_{c_i \in C_k, i \neq j} \mathbf{1}\{\alpha_i^{(j)} \leq \alpha_j^{(i)}\}}{|C_k| - 1}$;
// Adapted radius and growth
9 $r = \gamma(C_k)$;
// Radius adaption function
10 **for** $c_i \in C_k$ **do**
11 **for** $c_j \in S \backslash C_k$ **do**
12 **if** $p(c_i) \geq \epsilon$ *and* $d(c_i, c_j) \leq \delta$ **then**
13 $C_k = C_k \cup \{c_i\}$;
14 **until** *There is no $c \in S \backslash C_k$ added into C_k*;
15 $k \rightarrow k + 1$;
16 $S = S \backslash C_k$;

3 CONFORMAL DBSCAN

In this section, we propose *Conformal DBSCAN*, which combines DBSCAN with the conformal prediction framework. We first present the algorithm design and then extend the conformal prediction framework to provide theoretical analysis for our algorithm. Last, we discuss the relation with DBSCAN and provide a mathematical explanation for DBSCAN.

3.1 Algorithm

Given a collection of data objects $S = \{c_1, \cdots, c_n\}$, the function $d(c_i, c_j)$ describes the difference between two data objects c_i and c_j. Notice that this function does not need to be a metric function. We provide the pseudo-code of our conformal DBSCAN algorithm in Algorithm 1. Our algorithm proceeds in three phases.

Initial Members. Similar to DBSCAN, we find the high-density region as the initial members of a cluster. Given δ, which is the initial radius for initial seeds, we find the point x with the highest number of points within distance δ. If we cannot find such a point x (i.e., δ is too small) we increase δ until we can find one. Then this object and its neighbors are selected as the initial member of a cluster. This step is in Line 3-5 of Algorithm 1.

Core Objects Determination. Based on the current cluster members, we include new elements in an iterative manner. In each iteration, we first define the core objects as those whose conformity score is larger than a pre-specified significance level $\epsilon \in (0, 1)$. A smaller value ϵ results in a larger number of core objects. Generally, we set ϵ as 5% or 10%. In Algorithm 1 lines 7-8 calculate the discrepancy scores and conformity scores. The first condition in Line 12 is to determine the core objects.

Adapted Radius and Growth. Based on the core objects in the current cluster, whose discrepancy scores are ranked in the appropriate range, we include new objects within proximity from the core objects. Our algorithm adapts the radius according to the current cluster, represented by the function $r = \gamma(C)$, which depends on the current core objects, e.g., density, average distance and the maximum distance between any pair of closest objects in C. Then, all the objects within distance r from the core objects are included in the cluster, as shown in Line 9-13 of Algorithm 1. We will show in the next subsection that these newly included objects have a similar discrepancy score and they are believed to belong to this cluster with a high significance level.

Implementation. To apply this algorithm, the most important thing is to decide on the discrepancy score $\mathcal{A}$ and the adapted radius $r = \gamma(C)$. If we have some intuitions about the cluster structure, one can apply a data-dependent function as the discrepancy score. Otherwise, the k-nearest neighbors method is a good choice. The adapted radius depends on the distribution of objects in the clusters. Our method starts from the high-density region. Thus, the adapted radius is increased as the cluster grows. In our implementation, we take a data-driven method for $r = \gamma(C)$. For each object in the current cluster, we compute the distance to its k nearest neighbor and the largest distance is set as the radius r. This radius is recalculated when C grows and adapts to the property of the current cluster.

Suppose the discrepancy score is the sum of the distance to k nearest neighbors in the reference set. When a new data point is added in the cluster, we need to compute its discrepancy score and update the other existing objects' discrepancy scores. We can maintain a min heap for each data point to store the k smallest distance. Each data point needs to compute the discrepancy score once in $O(n \log k)$ time, where n is the number of objects. Then we can use the hash sort to select the core set, which is done in linear time.

Regarding the asymptotic behavior on n, the number of trajectories, DBSCAN has a running time of $O(n^2)$ and conformal DBSCAN has running time $O(n^2 \log k)$, where the extra factor, $\log k$, comes from using the min-heap to compute the discrepancy scores. In implementation we take k as a small constant thus $\log k$ can be skipped in the asymptotic running time.

3.2 Theoretical Analysis

In our iterative algorithm, we grow the cluster C one object at a time. Since we start with points in high-density regions (i.e., represented by a truncated distribution), the new point c_{m+1} may not follow the truncated distribution that characterizes the current reference set C. Recall that c_{m+1} is within distance r from a point in the core set of C, we denote by $\beta = \beta(C, r)$ the upper bound on the difference of the discrepancy scores of c_{m+1} and points of C within distance r.

Definition 3.1. *If a new object c_{m+1} and an object $c_i \in C$ are within distance r of each other, define the upper bound on the difference between their discrepancy scores based on the reference set $C^{-(i,j)} =$*

$C \backslash \{c_i, c_j\}$ *by* $\beta = \beta(C, r)$*:*

$$|\mathcal{A}(c_{m+1}, C^{-(i,j)}) - \mathcal{A}(c_i, C^{-(i,j)})| \leq \beta. \tag{4}$$

$\beta = \beta(C, r)$ *may depend on the reference set* C *and the radius* r*.*

It suggests that the discrepancy scores of two objects in a r-neighborhood, with respect to remaining points in C, are controlled with an upper bound β. For instance, suppose a cluster is formed by random draws from a certain distribution, for the sake of illustration, say a Gaussian distribution. If the initial cluster and the reference set C contain points in the high-density region, then the points in C are from a corresponding truncated Gaussian distribution. The new object c_{m+1} in the neighborhood of C and some points at or close to the boundary of C will be very similar. The discrepancy score of c_{m+1} and the scores of these close-by boundary points, with respect to the remaining points of C, will not be too far and have an upper bound. It provides a nice feature for the growth process that we do not actually need to calculate the discrepancy score of all the new data objects.

With this definition we modify the conformal prediction framework to handle the truncated distribution. Suppose, for a new object c_{m+1}, there exists a random draw $\tilde{c} \in C$ such that c_{m+1} and $\tilde{c}$ are in a r-neighborhood. We define a modified conformal score of c_{m+1}:

$$\tilde{p}(c_{m+1}) = 2\min\{\tilde{q}^-(c_{m+1}), \tilde{q}^+(c_{m+1})\} \tag{5}$$

where

$$\begin{aligned} \tilde{q}^-(c_{m+1}) &= \frac{\sum\limits_{i:c_i \neq \tilde{c}, 1 \leq i \leq m} \mathbf{1}\{\tilde{\alpha}_{m+1}^{(i)} \leq \tilde{\alpha}_i + \beta\}}{m-1} \\ \tilde{q}^+(c_{m+1}) &= \frac{\sum\limits_{i:c_i \neq \tilde{c}, 1 \leq i \leq m} \mathbf{1}\{\tilde{\alpha}_{m+1}^{(i)} \geq \tilde{\alpha}_i - \beta\}}{m-1} \end{aligned} \tag{6}$$

and $\tilde{\alpha}_{m+1}^{(i)} = \mathcal{A}(c_{m+1}, C \setminus \{c_i, \tilde{c}\})$ and $\tilde{\alpha}_i = \mathcal{A}(c_i, C \setminus \{c_i, \tilde{c}\})$, for $i \in \{i : c_i \neq \tilde{c}\}$. Then we have the following theorem.

Theorem 3.1. *Under the setting discussed above, we have*

$$\Pr(\tilde{p}(c_{m+1}) \leq \epsilon) \leq \epsilon.$$

Thus, the type-I error (i.e., the probability of not identifying c_{m+1} *as a sample in the cluster when in fact* c_{m+1} *is in the cluster) is less than* ϵ*.*

Proof. Without loss of generality, suppose $\tilde{c} = c_1$. We define $\tilde{C} = C \setminus \{\tilde{c}\} = \{c_2, \ldots, c_m\}$. Since $\tilde{c} = c_1$ and c_{m+1} are in a δ-neighborhood, by Definition 3.1 we have,

$$|\tilde{\alpha}_{m+1}^{(i)} - \tilde{\alpha}_1^{(i)}| \leq \beta, \text{ for } 2 \leq i \leq m \tag{7}$$

where $\tilde{\alpha}_{m+1}^{(i)} = \mathcal{A}(c_{m+1}, \tilde{C} \setminus \{c_i\})$ and $\tilde{\alpha}_i = \mathcal{A}(c_i, \tilde{C} \setminus \{c_i\})$, for $i = 2, \ldots, m$.

Furthermore, $\tilde{c} = c_1$ is one of the random objects in the set C, so c_1 is conformal with objects in set $\tilde{C}$. By treating c_1 as a 'new' object and comparing it to reference set $\tilde{C}$, the regular conformity score defined in Equation (1) and (2), is

$$p(c_1) = 2\min\{q^-(c_1), q^+(c_1)\} \tag{8}$$

where

$$\begin{aligned} q^-(c_1) &= \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} < \tilde{\alpha}_i\}}{m-1} + \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} = \tilde{\alpha}_i\}}{2(m-1)} \\ q^+(c_1) &= \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} > \tilde{\alpha}_i\}}{m-1} + \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} = \tilde{\alpha}_i\}}{2(m-1)} \end{aligned} \tag{9}$$

It is followed by Equation (3) that $\Pr(p(c_1) \leq \epsilon) \leq \epsilon$.

Now, we check the modified conformal score $\tilde{p}(c_{m+1})$. We first compare the relationship between $\tilde{q}^-(c_{m+1})$ and $q^-(c_1)$.

$$\begin{aligned} \tilde{q}^-(c_{m+1}) &= \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_{m+1}^{(i)} \leq \tilde{\alpha}_i + \beta\}}{m-1} \\ &= \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} - \tilde{\alpha}_i \leq \tilde{\alpha}_1^{(i)} + \beta - \tilde{\alpha}_{m+1}^{(i)}\}}{m-1} \\ &\geq \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} - \tilde{\alpha}_i \leq 0\}}{m-1} \\ &= \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} \leq \tilde{\alpha}_i^{(1)}\}}{m-1} \\ &\geq \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} < \tilde{\alpha}_i\}}{m-1} + \frac{\sum_{i=2}^m \mathbf{1}\{\tilde{\alpha}_1^{(i)} = \tilde{\alpha}_i\}}{2(m-1)} \\ &= q^-(c_1) \end{aligned}$$

The variable $\tilde{\alpha}_1^{(i)}$ is inserted on the both sides on Line 3. According to Equation (7), the right side of Line 3 is not less than 0 and Line 4 is obtained. Similarly, we can get $\tilde{q}^+(c_{m+1}) \geq q^+(c_1)$. It leads to

$$\begin{aligned} \tilde{p}(c_{m+1}) &= 2\min\{\tilde{q}^-(c_{m+1}), \tilde{q}^+(c_{m+1})\} \\ &\geq 2\min\{q^-(c_1), q^+(c_1)\} \\ &= p(c_1) \end{aligned} \tag{10}$$

then, we can conclude this theorem, i.e.,

$$\Pr(\tilde{p}(c_{m+1}) \leq \epsilon) \leq \Pr(p(c_1) \leq \epsilon) \leq \epsilon$$

□

Based on this theorem, when the object c_{m+1} is close enough to one of the randomly selected conformal objects in the reference set, its discrepancy score is bounded. Then, its corresponding modified conformity score is also controlled. If we reject the neighboring c_{m+1} to be included in the cluster when its modified conformity score $\tilde{p}(c_{m+1}) < \epsilon$, then the type-I error of this labeling is controlled to be less than ϵ.

Theorem 3.1 holds for any bound β in Definition 3.1. If β is too large the detecting rule based on the corresponding $\tilde{p}(c_{m+1})$ can be overly conservative, but the way Algorithm developed allows us to define the modified conformity score $\tilde{p}(c_{m+1})$ used in our detection rule as the one computed using the tightest bound that satisfies Definition 3.1.

3.3 Relation with DBSCAN

Conformal DBSCAN can be considered as a generalization and improvement to vanilla DBSCAN. In both DBSCAN and conformal DBSCAN we define core points and include new points by proximity to these core points. Suppose we define the discrepancy score for x as the distance to the $(minPts)$th nearest neighbor in the reference set C. For DBSCAN, the core points have discrepancy scores of less

Dataset	Synthetic	Animal	Human
# Agents	260	102	633,194
# Clusters	5	3	Varying
# Records	4,160	14,990	50,873,192
Avg # Records/Agent	16	147	80

Table 1: Dataset Description

than r. For conformal DBSCAN, the core objects have conformity score $p(c)$ larger than ϵ. The directly-reachable objects in DBSCAN are similar to the objects whose modified conformity score $\tilde{p}(c)$ are larger than ϵ.

4 EVALUATION BY SIMULATION

In this section, we present experimental evaluations of conformal DBSCAN algorithm for trajectory clustering. Section 4.1 is about the setup of our experiments including datasets and baseline algorithms as references. The clustering performance is discussed in Section 4.2. We also presented the application of conformal prediction framework for supervised classification tasks in Section 4.3. In Section 4.4, we provide some discussions and observations on trajectory clustering, classification with respect to sampling rate and similarity measures.

4.1 Experimental Setup

Hardware. We implemented our algorithm in Python with version 3.8. We ran the experiments on the machine equipped with Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, and 32GB of RAM.

Dataset. We used three trajectory datasets, with different collection methods and features. The description of them is shown in Table 1 and the details are introduced below:

(1) Synthetic Trajectory: A public dataset[1] of simulated trajectories was previously created by Piciarelli et al. [32]. It consists of 1,000 randomly generated datasets. Each of these datasets contains 260 2-dimensional trajectories of length 16, i.e., exactly 16 points. There are 5 different clusters, and each one contains 250 normal trajectory. The remaining 10 are stray trajectories that do not belong to any cluster. Figure 3 shows three cases of this dataset.

(2) Animal Trajectory: The animal movement dataset[2] was generated by the Starkey project. This dataset contains the radio-telemetry location (with other information) of elk, deer, and cattle from spring through fall for the year 1993 through 1996. It includes more than 287,000 observations acquired from animals in a natural setting. We extract the coordinates, as well as the record time information, from the telemetry data observed in June 1995, shown in Figure 4(a). The trajectories can be divided into three classes by species. In the Elk category there are 38 trajectories and 7,117 data points; in the deer category there are 30 trajectories and 4,333 data points; and in the cattle category there are 34 trajectories and 3,540 data points. Therefore, there are on average 147 points for each animal over one month.

(3) Human Trajectory: The trajectories are collected from electric motorbikes in Wenzhou, China. The datasets include a list of points with vehicle ID, system time, longitude, and latitude. For each vehicle, there are 30-days trajectories in June 2018. The location on the trajectory is taken as the location of the checkpoints that have recorded such an appearance. There are 5,228 checkpoints in the city with an area $110km \times 70km$. A total of 633,194 trajectories are recorded and each person has about 80 points on average in one day. In the experiment, we randomly select a subset of vehicles and run clustering for all trajectories of these vehicles.

Baseline Algorithms. For trajectory clustering, we have implemented the classical unsupervised clustering algorithms, k-means, DBSCAN and DBSCAN-GM [2], to obtain the baseline results. The distance between trajectories is defined as 1 minus their similarity. We use three similarity measures proposed in recent work [42]: Time-Sensitive Similarity (TSS), Order-Sensitive Similarity (OSS) and Order-Insensitive Similarity (OIS). Note that these three similarity measures are non-metric functions. We also use two classical similarity measures, Hausdorff distance and Fréchet distance. A brief introduction of these similarities is shown:

(1) **Time-Sensitive Similarity**: Each trajectory is a sequence of time-stamped locations visited by the agent. Two trajectories are α-Time-Sensitive similar if at least α fraction of samples on the shorter trajectory are the same ones (or nearby with a specified distance/time threshold) on the other trajectory, including time stamps and locations.
(2) **Order-Sensitive Similarity**: Each trajectory is an ordered sequence of locations visited by the agent. Two trajectories are α-order-sensitive similar, if an α fraction of samples on the shorter trajectory are matched with the samples on the other trajectory in the corresponding order. This could be considered as a partial measure for Fréchet distance.
(3) **Order-Insensitive Similarity**: Each trajectory is a set of visited locations. Two trajectories are α-Order-Insensitive similar if at least α fraction of locations on the shorter trajectory are matched with the other trajectory. It ignores the time dimension and visiting order issue. This could be considered as a partial measure for Hausdorff distance.
(4) **Hausdorff Distance**: Let X and Y be two non-empty subsets of a metric space (M, d). Their Hausdorff distance $d_H(X, Y)$ is defined as

$$d_H(X, Y) = \max\{\max_{x \in X} \min_{y \in Y} d(x, y), \max_{y \in Y} \min_{x \in X} d(x, y)\}$$

where $d(x, y)$ quantifies the distance between points $x \in X$ and $y \in Y$.
(5) **Fréchet Distance**: Let $f : [0, m] \to \mathbb{R}^k$ and $g : [0, n] \to \mathbb{R}^k$ be two polygonal curves or sequences. The Fréchet distance is defined as

$$d_F(f, g) = \min_{\alpha, \beta} \max_{s \in [1, m+n]} \{d(f(\alpha(s)), g(\beta(s)))\}$$

where α and β range over all discrete non-decreasing onto mappings of the form $\alpha : [1 : m + n] \to [0 : m]$, $\beta : [1 : m + n] \to [0 : n]$.

Evaluation Metric. To evaluate the classification results, we use precision and recall to measure the performance. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while the recall (also known as

[1] http://avires.dimi.uniud.it/papers/trclust/
[2] http://www.fs.fed.us/pnw/starkey/data/tables/

sensitivity) is the fraction of the total number of relevant instances that are retrieved.

For the trajectory clustering tasks, adjusted for chance measures are widely used to compare partitions/clustering of the same dataset. The Rand Index (RI) computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. It can be viewed as a measure of the percentage of correct decisions made by the algorithm, computed by

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. The Adjusted Rand Index (ARI) is the corrected-for-chance version of the Rand Index. It is thus ensured to have a value close to 0 for random labeling independently of the number of clusters and samples, and exactly 1 when the clusterings are identical (up to a permutation). In addition, the Adjusted Mutual Information (AMI) based on Shannon information theory is another popular metric in the clustering community. Simone et al. [34] proposed that ARI should be used when the reference clustering has large equal size clusters; AMI should be used when the reference clustering is unbalanced and there exist small clusters. Our three trajectory datasets are much suitable for the ARI metric, as our evaluation metric, because all the clusters in our datasets have a similar size.

Evaluation Step. We implemented the proposed algorithms based on Python and our source code are publicly shared on Github [1]. We provide a tool-chain of our framework, including data cleaning, similarity computation and clustering algorithms. Here are the main steps in our tool-chain:

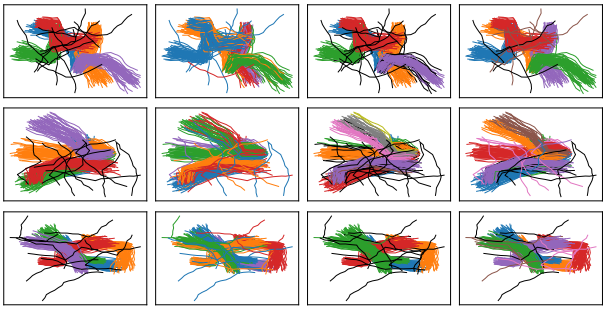
(1) Data Preprocessing: For the human trajectory dataset, the sampling rate is about 10 seconds. There are many outliers in the trajectories, such as unrealistic speed or impossible locations. We need to clean these human trajectory data to make them realistic.

(2) Similarity Computation: For three partial similarity measures, the corresponding efficient algorithms are introduced in the previous work [42]. When the number of data points and the number of trajectories are large, we could use uniform sampling to obtain an approximation of the partial similarity measure, to reduce computational time.

(3) Clustering Algorithm: Based on the similarity matrix, we implemented k-means, DBSCAN and conformal DBSCAN to output the clustering results. Compared with the ground truth, we analyze the performance (ARI, recall, and precision).

4.2 Trajectory Clustering

First, we test our algorithm under the unsupervised settings without any prior knowledge. Thus, for the k-means algorithm, we use the "elbow" method to select the optimal number of clusters. It plots the explained variance as a function of the number of clusters and picks the elbow of the curve as the number of clusters to use. For DBSCAN, all reasonable parameters combinations (the radius of neighborhoods and the minimum number of points within radius) are enumerated to find the best performance. In our algorithm, we use the sum distance to k nearest neighbors as the discrepancy score.



(a) Ground Truth (b) k-means (c) DBSCAN (d) Conformal DBSCAN

Figure 3: Clustering for synthetic trajectories: The first column is the ground truth of three cases. 250 trajectories belong to 5 clusters in different colors, and the outliers are in black. The clustering results by k-means, DBSCAN, Conformal DBSCAN are shown in the following columns.

Method	k-Means	DBSCAN	Conformal DBSCAN
OSS	0.5792	0.8703	**0.8970**
OIS	0.5774	0.8196	**0.8507**

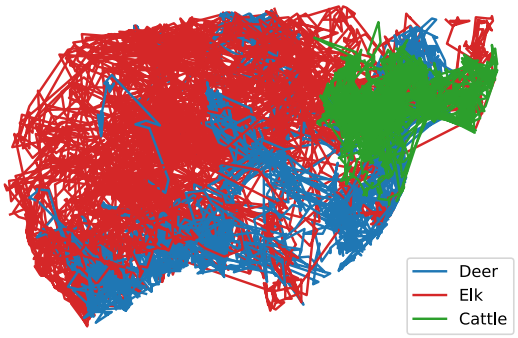
Table 2: Clustering result for synthetic trajectories

For any pair of trajectories within the difference r for these three similarity measures, the bound β in Definition 3.1 can be written as the function $\beta(C, r) = \frac{kr}{1-r}$. The radius is adjusted with the largest $[\frac{k}{2}]$-nearest neighbor distance to include more trajectories. Then the performance of three datasets are shown below.

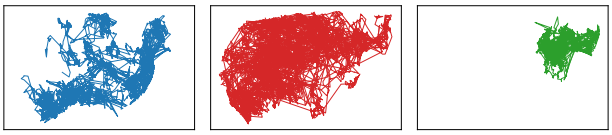
Synthetic Trajectory. Given that trajectories are generated without specific time information, we only compute the order-sensitive similarity and order-insensitive similarity between trajectories. There are three cases are shown in Figure 3. In Figure 3(a), the distribution of raw trajectories is demonstrated, including normal trajectories in 5 clusters with different colors and outliers (black). The ground truth takes the normal trajectories in five clusters and one cluster for all outliers.

The clustering results by k-means algorithm are shown in Figure 3(b). We can see that the size of clusters are not even. Some trajectories in the same reference cluster are grouped into different clusters, and some in the different reference clusters are mixed. In addition, it cannot distinguish the outlier trajectories. Figure 3(c) shows the clustering results by DBSCAN. It is much better than the results of k-means. However, we can still find misplaced trajectories, in the second and third case. The good thing is that outlier trajectories can be distinguished. The clustering results by conformal DBSCAN algorithm are shown in Figure 3(d). In general our results are better than the result of DBSCAN. Some outlier trajectories are put in a separate cluster.

The overall performance on 1,000 trajectories using two similarity measures is shown in Table 2. It is obvious that the results of conformal DBSCAN algorithm are the best, with a higher ARI value. In addition, the performance using the order-sensitive similarity is better than that of using the order-insensitive similarity, because

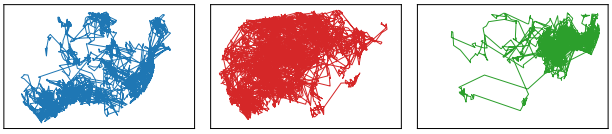


(a) Trajectories of all animal species



(b) Reference cluster of each animal specie

(c) Cluster results of each animal specie by DBSCAN



(d) Cluster results of each animal specie by conformal DBSCAN

Figure 4: Clustering results for animal trajectories. (a) and (b) are the trajectories of three species of animals (Red: Elk, Blue: Deer, and Green: Cattle). (c) and (d) show the clustering results by DBSCAN and conformal DBSCAN algorithm.

the traversal order is a good factor not only for separating different clusters, but also enlarging the difference between trajectories in one cluster in this dataset. In the following experiment, we will focus on the comparison between DBSCAN and conformal DBSCAN algorithm.

Animal Trajectory. Figure 4(a) shows all the trajectories of these three animal species in one month. Their activity regions partially overlapped. The trajectory distribution of each species, i.e., ground truth, is shown in Figure 4(b).

Here, we use time-sensitive similarity to compare two animals' trajectories. We notices that each animal has about 15 data points per trajectory trace, but these data points are not recorded evenly. Thus, we set the time slot as one day, which means that if two animals visit the same region on the same day, it should be counted into the similarity measure. First, the clustering result by DBSCAN is shown in Figure 4(c). The trajectories of deer and elk are not separated. While using conformal DBSCAN algorithm, the clustering results are much better in Figure 4(d), separating deer and elk. The

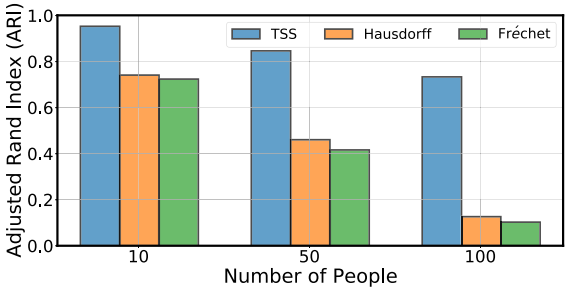


Figure 5: Clustering results using Hausdorff distance and Fréchet distance

# Trajectories	200	500	1,000	2,000
DBSCAN (ms)	3.242	23.119	97.401	398.054
Conformal DBSCAN (ms)	5.312	38.798	173.452	711.248

Table 3: The running time of DBSCAN and conformal DBSCAN for human trajectories

ARI value for DBSCAN is 0.6020, while the ARI for our algorithm is 0.7449.

Human Trajectory In this dataset, each person has about 30 daily trajectories, representing regular daily routine. We randomly select a subset of subjects and retrieve all the trajectories of these subjects from the dataset. Our goal is to separate all the trajectories into clusters, each belonging to a single subject.

First, the performance of using Hausdorff distance and Fréchet distance as the similarity measures is shown in Figure 5. In terms of the ARI score, conformal DBSCAN using Hausdorff distance and Fréchet distance performs much worse than that using Time-Sensitive similarity especially when the number of agents in the dataset is increased. The reason is that even for the same subject, her trajectories in two days have a large Hausdorff distance and Fréchet distance, making it hard to group trajectories of the same person.

We vary the number of subjects from 10 to 200, and show results in Figure 6. It shows the ARI value of clustering by DBSCAN and conformal DBSCAN algorithm using three similarity measurements. First, with the increasing number of subjects, the ARI value is reduced. This is natural, as it is more likely to have similar trajectories from different subjects when the number of subjects increases, making it harder to separate the subjects. We made an assumption that trajectories of the same individual are separable. This assumption is more likely to break down as there are more subjects. When there are 200 people selected, DBSCAN can only generate about 60 clusters and conformal DBSCAN algorithm gets about 130 clusters, which is a significant improvement.

Second, using the same similarity measure, conformal DBSCAN outperforms DBSCAN and DBSCAN-GM [2], especially when the number of subjects is large. The reason is that DBSCAN has a predetermined and fixed radius to include trajectories. It is hard to select an appropriate radius manually. Although the DBSCAN-GM

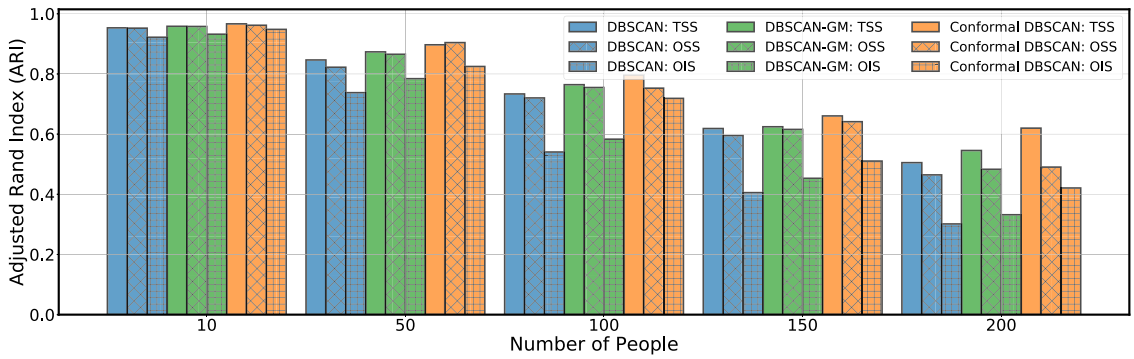


Figure 6: Clustering results of DBSCAN, DBSCAN-GM and conformal DBSCAN algorithm using different similarity measures

algorithm combines the Gaussian-Mean method to select the centers and radius, this radius is not perfect due to variation in cluster density. Subjects vary significantly in terms of the similarity of their trajectories. Conformal DBSCAN can adapt the radius according to clusters' growing process. It performs better in the cases with clusters of varying density.

In addition, the same clustering algorithm shows different performance with three similarity measures. Clustering with time-sensitive similarity performs better than that with order-sensitive and order-insensitive similarity. This suggests that the traversal order and time information can enlarge the gap between the similarity of the same subject and the similarities of two different subjects, which improves clustering accuracy.

Running time. Table 3 shows the running time of two methods, DBSCAN and conformal DBSCAN, after the similarity matrix is obtained. As suggested by the theoretical analysis, the running time of conformal DBSCAN is only modestly higher than the classical DBSCAN with essentially the same scaling behavior in the number of trajectories.

4.3 Trajectory Classification

In this section, we present the application of conformal prediction framework for (supervised) classification problem. For each dataset, we select a part of trajectories as the reference set, and the rest as the test set. The discrepancy score is the sum distance to k nearest neighbors. With different significance levels ϵ, we compute the conformity score for a test trajectory with respect to each reference cluster. If the conformity score is larger than ϵ, it will carry the cluster label. In this way, one trajectory might have multiple labels and we check the precision and recall metrics.

Synthetic Trajectory. For each cluster of 50 trajectories, we randomly choose 20 trajectories as the reference cluster. The rest and the outlier trajectories are the test set. The significance level ϵ is set as 0.05, 0.1 and 0.15, respectively. The classification results with order-sensitive and order-insensitive similarity are shown in Table 4.

First, the recall value is higher than $1 - \epsilon$. It is guaranteed by the framework that the prediction set contains the true cluster label with $(1 - \epsilon)100\%$ significance level. Then, with a higher significance level ϵ, the precision value is increased. The reason is that the prediction set becomes smaller, so the fraction of relevant trajectories among retrieved trajectories becomes higher. In addition, using order-sensitive similarity has a better precision than the order-insensitive similarity, because considering traversal order enlarges the gap between different clusters, reducing the number of false positive instances.

Similarity	OSS			OIS		
ϵ	5%	10%	15%	5%	10%	15%
Precision	0.8067	0.9385	0.9652	0.7329	0.9021	0.9455
Recall	0.9617	0.9237	0.8842	0.9617	0.9237	0.8842

Table 4: Classification result of synthetic trajectories

Human Trajectory. For each subject, 15 daily trajectories are taken as the reference set, with which we wish to successfully identify the other 15 trajectories of the same subject. Given a significance level $\epsilon = 0.1$, we check the number of subjects from 10 to 200 with three similarity measures. The recall value is about 0.82, a little lower than the expected result 0.9. The reason is that 30 daily trajectories of one person in the ground truth are probably not a sufficiently representative cluster in practice.

The precision results are shown in Figure 7. With the number of subjects increasing, the precision result using time-sensitive similarity is reduced substantially, while the precision with other measures still remains high (above 0.7). The Order-Sensitive Similarity and Order-Insensitive Similarity seem to be good for this task when the number of subjects is above 150. The reason is that the variance of the discrepancy scores of conformal trajectories are large, making it easy for the test trajectory to get a comparable discrepancy score in different reference sets.

On the other hand, with trajectories of 200 agents, if each trajectory is simply labeled by the label of the reference cluster with highest conformity score, the accuracy is still reasonable (around 60% – i.e., 60% trajectories are assigned the correct label) .

4.4 Discussion

Our experiments show that the three partial similarity measures have respective merits in different tasks and datasets. The pattern is persistent when we reduce the sampling rate. Similar clustering

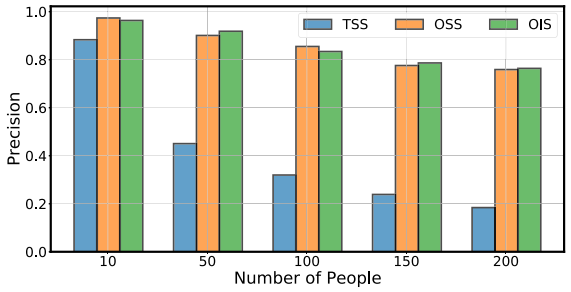


Figure 7: Precision results for human trajectory classification with different similarity measures.

results are obtained. Thus, it is important to choose appropriate measures to analyze the mobility trajectories.

With technology innovation, trajectory collection efforts often strive for fine-grained mobility data with accurate localization and dense sampling intervals. High sampling rate increases computation time. The experiments carried out here do not observe significant benefit with increases in sampling rate, for the purpose of recognizing trajectories of different individuals.

Similarly, it is tempting to incorporate more features, in order to improve the performance of clustering and classification tasks. Our experimental results suggest that we should take this with caution. The main factor is to recognize the gap between similarity of two trajectories within the same clusters and across different clusters. With an increasing number of participants, the performance of time-sensitive similarity actually drops, while the performance of order sensitive and order insensitive similarities remain high. This suggests that the time-sensitive information could be too detailed to recognize mobility patterns of different individuals.

5 CONCLUSION

In this paper, we proposed an algorithm combining DBSCAN algorithm and the conformal prediction framework. Conformal DBSCAN improves DBSCAN algorithm by automatically adapting the parameters to the data being handled, guided by theoretical analysis of an improved conformal prediction framework. Conformal DBSCAN shows superior performance on a variety of artificial and real-world trajectory datasets. We remark that conformal DBSCAN is a generic unsupervised clustering algorithm that has potential on other unstructured, messy data. In future work we plan to test conformal DBSCAN on other datasets, by incorporating more sophisticated discrepancy scores such as those obtained from deep learning encoders.

Acknowledgement The authors would like to acknowledge supports from NSF DMS-2015373, NSF DMS-2027855, NSF DMS-1812048, NSF OAC-1939459 and NSF CCF-2118953.

REFERENCES

[1] Code. https://github.com/SBUhaotian/Conformal_DBSCAN.
[2] Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan. 2007), 36–44. https://doi.org/10.1145/1188913.1188915
[3] Gianluca Antonini and Jean-Philippe Thiran. 2006. Counting pedestrians in video sequences using trajectory clustering. *IEEE Transactions on Circuits and Systems for Video Technology* 16, 8 (2006), 1008–1020.
[4] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. 2019. Predictive inference with the jackknife+. *arXiv preprint arXiv:1905.02928* (2019).
[5] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and Francois Royer. 2017. Destination prediction by trajectory distribution-based model. *IEEE Transactions on Intelligent Transportation Systems* 19, 8 (2017), 2470–2481.
[6] Meng Chen, Yang Liu, and Xiaohui Yu. 2015. Predicting next locations with object clustering and trajectory clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 344–356.
[7] X. Chen, W. Liu, H. Qiu, and J. Lai. 2011. APSCAN: A parameter free algorithm for clustering. *PATTERN RECOGNITION LETTERS* (2011).
[8] D Devetyarov and I Nouretdinov. 2008. Prediction with confidence based on a random forest classifierlearning for medical diagnosis. In *Proceedings of 6th IFIP WG*, Vol. 12. 37–44.
[9] Gal Elidan. 2013. Copulas in machine learning. In *Copulae in mathematical and quantitative finance*. Springer, 39–60.
[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[11] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. 2020. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6797–6806.
[12] Ana Fred and Anil K Jain. 2002. Evidence accumulation clustering based on the k-means algorithm. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 442–451.
[13] Andre Salvaro Furtado, Laercio Lima Pilla, and Vania Bogorny. 2018. A branch and bound strategy for Fast Trajectory Similarity Measuring. *Data & Knowledge Engineering* 115 (2018), 16–31.
[14] Min ge Xie and Zheshi Zheng. 2022. Homeostasis phenomenon in conformal prediction and predictive distribution functions. *International Journal of Approximate Reasoning* 141 (2022), 131–145. https://doi.org/10.1016/j.ijar.2021.09.001
[15] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* 38 (1985), 293–306.
[16] Jian Hou, Huijun Gao, and Xuelong Li. 2016. DSets-DBSCAN: A parameter-free clustering algorithm. *IEEE Transactions on Image Processing* 25, 7 (2016), 3182–3193.
[17] Weiming Hu, Xi Li, Guodong Tian, Stephen Maybank, and Zhongfei Zhang. 2013. An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE transactions on pattern analysis and machine intelligence* 35, 5 (2013), 1051–1065.
[18] D Huang and W. Peng. 2012. Grid-based DBSCAN Algorithm with Referential Parameters. *Physics Procedia* 24, part-PB (2012), 1166–1170.
[19] Arash Jahangiri and Hesham A Rakha. 2015. Applying machine learning techniques to transportation mode recognition using mobile phone sensor data. *IEEE transactions on intelligent transportation systems* 16, 5 (2015), 2406–2417.
[20] W. Lai, M. Zhou, F. Hu, K. Bian, and Q. Song. 2019. A New DBSCAN Parameters Determination Method Based on Improved MVO. *IEEE Access* 7 (2019), 104085–104095.
[21] Antonis Lambrou, Harris Papadopoulos, and Alex Gammerman. 2009. Evolutionary conformal prediction for breast cancer diagnosis. In *2009 9th international conference on information technology and applications in biomedicine*. IEEE, 1–4.
[22] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 593–604.
[23] Qingquan Li, Zhe Zeng, Bisheng Yang, and Tong Zhang. 2009. Hierarchical route planning based on taxi GPS-trajectories. In *17th International Conference on Geoinformatics*. IEEE, 1–5.
[24] Xi Li, Weiming Hu, and Wei Hu. 2006. A coarse-to-fine strategy for vehicle motion trajectory clustering. In *18th International conference on pattern recognition (ICPR'06)*, Vol. 1. IEEE, 591–594.
[25] Xiaojiang Li, Paolo Santi, Theodore K Courtney, Santosh K Verma, and Carlo Ratti. 2018. Investigating the association between streetscapes and human walking activities using Google Street View and human trajectory data. *Transactions in GIS* 22, 4 (2018), 1029–1044.
[26] Yang Li, Qixing Huang, Michael Kerber, Lin Zhang, and Leonidas Guibas. 2013. Large-scale joint map matching of GPS traces. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 214–223.
[27] Weiyao Lin, Yang Zhou, Hongteng Xu, Junchi Yan, Mingliang Xu, Jianxin Wu, and Zicheng Liu. 2016. A tube-and-droplet-based approach for representing and analyzing motion trajectories. *IEEE transactions on pattern analysis and machine intelligence* 39, 8 (2016), 1489–1503.
[28] Shengzhong Liu, Shuochao Yao, Xinzhe Fu, Huajie Shao, Rohan Tabish, Simon Yu, Ayoosh Bansal, Heechul Yun, Lui Sha, and Tarek Abdelzaher. 2021. Real-Time Task Scheduling for Machine Perception in In Intelligent Cyber-Physical Systems. *IEEE Trans. Comput.* (2021), 1–1. https://doi.org/10.1109/TC.2021.3106496

[29] Brendan Morris and Mohan Trivedi. 2009. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 312–319.

[30] Harris Papadopoulos, Volodya Vovk, and Alex Gammerman. 2007. Conformal prediction with neural networks. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, Vol. 2. IEEE, 388–395.

[31] José M Pena, Jose Antonio Lozano, and Pedro Larranaga. 1999. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters* 20, 10 (1999), 1027–1040.

[32] Claudio Piciarelli, Christian Micheloni, and Gian Luca Foresti. 2008. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for video Technology* 18, 11 (2008), 1544–1554.

[33] Moonsoo Ra, Chiawei Lim, Yong Ho Song, Jechang Jung, and Whoi-Yul Kim. 2015. Effective trajectory similarity measure for moving objects in real-world scene. In *Information Science and Applications*. Springer, 641–648.

[34] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research* 17, 1 (2016), 4635–4666.

[35] Govind Salvi. 2014. An automated nighttime vehicle counting and detection system for traffic surveillance. In *2014 International Conference on Computational Science and Computational Intelligence*, Vol. 1. IEEE, 131–136.

[36] Abir Smiti and Zied Elouedi. 2012. Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques. In *2012 IEEE 16th international conference on intelligent engineering systems (INES)*. IEEE, 573–578.

[37] N. Soni and A. Ganatra. 2016. AGED (Automatic Generation of Eps for DBSCAN). In *International Journal of Computer Science and Information Security (IJCSIS)*.

[38] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal* 29, 1 (2020), 3–32.

[39] Na Ta, Guoliang Li, Yongqing Xie, Changqi Li, Shuang Hao, and Jianhua Feng. 2017. Signature-based trajectory similarity join. *IEEE Transactions on Knowledge and Data Engineering* 29, 4 (2017), 870–883.

[40] Mohammed M Vazifeh, Paolo Santi, Giovanni Resta, Steven H Strogatz, and Carlo Ratti. 2018. Addressing the minimum fleet problem in on-demand urban mobility. *Nature* 557, 7706 (2018), 534–538.

[41] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer Science & Business Media.

[42] Haotian Wang and Jie Gao. 2020. Distributed Human Trajectory Sensing and Partial Similarity Queries. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 253–264.

[43] Wei Wang, Feng Xia, Hansong Nie, Zhikui Chen, Zhiguo Gong, Xiangjie Kong, and Wei Wei. 2020. Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[44] Ruizhi Wu, Guangchun Luo, Junming Shao, Ling Tian, and Chengzong Peng. 2018. Location prediction on trajectory data: A review. *Big data mining and analytics* 1, 2 (2018), 108–127.

[45] Minge Xie and Zheshi Zheng. 2020. Discussion of Professor Bradley Efron's Article on, "Prediction, Estimation, and Attribution". *J. Amer. Statist. Assoc.* 115, 530 (2020), 667–671.

[46] Hongteng Xu, Yang Zhou, Weiyao Lin, and Hongyuan Zha. 2015. Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage. In *Proceedings of the IEEE International Conference on Computer Vision*. 4328–4336.

[47] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2017. Trajectory clustering via deep representation learning. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 3880–3887.

[48] Huijing Zhao, Chao Wang, Yuping Lin, Franck Guillemard, Stephane Geronimi, and Francois Aioun. 2016. On-road vehicle trajectory collection and scene-based lane change analysis: Part i. *IEEE Transactions on Intelligent Transportation Systems* 18, 1 (2016), 192–205.

[49] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. 312–321.

[50] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*. 247–256.

[51] H. Zhou, P. Wang, and H. Li. 2012. Research on adaptive parameters determination in DBSCAN algorithm. *Journal of Xi'an University of Technology* (2012).