IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

Clustering the Movement of Trajectory Based Data

Bhupeshwar Kumar Sahu[1], Dr. Vishnu Mishra[2]

Ph.D (Research Scholar), Department of IT & CS, Dr. C.V. Raman University, Bilaspur, India [1]

Associate Professor, CSE, Bharti College of Engineering & Technology, Durg, India [2]

Abstract: This paper group the trajectories objects of moving data. Our observation that develops the new algorithms that manage Minimum Description Length (MDL). This paper based on k-nearest algorithm with LCSS model and dimensional Euclidean. Our algorithm consists of two parts that is partitioning and grouping phase. Experimental result searching the minimum distance by using MOTRACLUS algorithm and analysis the sub-trajectories and real-trajectories. The experimental analysis gives moving object of trajectory animals.

Keywords: Data mining, Density based cluster, Trajectory algorithm, Framework grouped, Partition and group work.

I. INTRODUCTION

In this paper we investigate the problem of discovering similar trajectories of moving objects. The trajectory of a moving object is typically modelled as a sequence of consecutive locations in a multidimensional (generally two or three dimensional) Euclidean space. Such data types arise in many applications where the location of a given object is measured repeatedly over time. Examples include features extracted from video clips, animal mobility experiments, sign language recognition, mobile phone usage, multiple attribute response curves in drug therapy, and so on. Moreover, the recent advances in mobile computing, sensor and GPS technology have made it possible to collect large amounts of spatiotemporal data and there is increasing interest to perform data analysis tasks over this data [4]. For example, in mobile computing, users equipped with mobile devices move in space and register their location at different time instants via wireless links to spatiotemporal databases. In environmental information systems, tracking animals and weather conditions is very common and large datasets can be created by storing locations of observed objects over time. Data analysis in such data includes determining and finding objects that moved in a similar way or followed a certain motion pattern. An appropriate and efficient model for defining the similarity for trajectory data will be very important for the quality of the data analysis task. Robust distance metrics for trajectories, In general these trajectories will be obtained during a tracking procedure, with the aid of various sensors. Here also lies the main obstacle of such data; they may contain a significant amount of outliers or in other words incorrect data measurements (unlike for example, stock data which contain no errors whatsoever. Our objective is the automatic classification of trajectories using Nearest Neighbor Classification. It has been shown that the one nearest neighbor rule has asymptotic error rate that is at most twice the Bayes error rate [12]. Previous approaches to model the similarity between time-series include the use of the Euclidean and the Dynamic Time Warping (DTW) distance, which however are relatively sensitive to noise. Distance functions that are robust to extremely noisy data will typically violate the triangular inequality. These functions achieve this by not considering the most dissimilar parts of the objects. However, they are useful, because they represent an accurate model of the human perception, since when comparing any kind of data (images, trajectories etc), we mostly focus on the portions that are similar and we are willing to pay less attention to regions of great dissimilarity. For this kind of data we need distance functions that can address the following issues: Different Sampling Rates or different speeds. The time-series that we obtain, are not guaranteed to be the outcome of sampling at fixed time intervals. The sensors collecting the data may fail for some period of time, leading to inconsistent sampling rates. Moreover, two time series moving at exactly the similar way, but one moving at twice the speed of the other will result (most probably) to a very large Euclidean distance.

II. RELATED WORK

In this Section, we study previous work based on two major discovery techniques, Markov chain models and spatiotemporal data mining, for extracting movement patterns of an object from historical trajectories. Markov chain models have been widely used in order to estimate the probability of an object's movements from one region or state to another at next time period. Ishikawa et al. derive the Markov transition probabilities between cells from indexed trajectories [1].

In their further study [7], a special type of histogram, called mobility histogram, is used to describe mobility statistics based on the Markov chain model. They also represent the histogram as cube-like logical structures and support an OLAP-style analysis. Authors in [8] classify an object's mobility patterns into three states (stationary state, linear

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

movement, and random movement) and apply Markov transition probabilities to explain a movement change one state to another. [9, 10] consider the location tracking problem in PCS networks.
Both studies are based on the same Markov process in order to describe users' movements from one or multiple PCS cells to another cell. However, they have different ways to model users' mobilities using Morkov models, thus, show distinct results to each other.

Spatiotemporal data mining methods have been also studied well for describing objects' patterns. [2] Introduces mining algorithms that detect a user's moving patterns, and exploits the mined information to invent a data allocation method in a mobile computing environment. Another mining technique is shown in [11]. This study focuses on discovering spatio-temporal patterns in environmental data. In [6], authors do not only explore periodic patterns of objects but also present indexing and querying techniques of the discovered or no discovered pattern information. [3, 4] address spatio-temporal association rules of the form $(r_i,t_1,p)\rightarrow (r_j,t_2)$ with an appearance probability p, where i and rj are regions at time (interval) t1 and t2 respectively ($t_2>t_1$). It implies that an object in r_i at time t_1 is likely to appear in r_j at time t_2 with p% probability. Besides, [3] considers spatial semantic areas (i.e. sources, sinks, stationary regions, and thoroughfares) in each r_i as well as more comprehensive definitions and algorithms of spatio-temporal association rules.

All above studies except [6] are based on the space-partitioning schemes, thus, the discovery accuracy depends on how the system decides space granularity of data space. When they partition the data space into a large number of small size cells, the accuracy increases, however, managing such many cells in memory can be burden to the system. On the contrary, using large size cells for partitioning cause low precision of discovery. Moreover, they cannot avoid the answer-loss problem no matter how the spatial granularity is set to.

III. PROPOSED ALGORITHM

The MOTRACLUS Algorithm

The moving object trajectory clustering algorithm MOTRACLUS consist of two phases. Its execute two algorithms to perform the subtasks (lines 2 and 4), in the first phase we execute the trajectory partitioning algorithm and then second phase we execute the trajectory clustering algorithm. We detailed explain these algorithms in Section 1 and 2.

Algorithm MOTRACLUS (Moving Object-Trajectory Clustering)

Input: A set of trajectories I = {TRJ1,⋯,TRJntr}
Output: (1) A set of clusters O = {*CL1*,⋯,*CLncls*}
(2) A set of representative trajectories

Algorithm:

```
    /* Partitioning Phase */
01: for each (TRJ ∈ I) do
02:     Execute Approximate Trajectory Partitioning;
          Gets a set L of line segments using the result;
03:       Accumulate L into a set D;
    /* Grouping Phase */
04:       Execute Relative Density-Based Clustering for D;
          Get a set of clusters as the result;
```

1. Trajectory Partitioning

In this section, we propose a trajectory partitioning algorithm for the partitioning phase. We first discuss two desirable properties of trajectory partitioning. We then describe a formal method for achieving these properties. Our method transforms trajectory partitioning to MDL optimization. Since the cost of finding the optimal solution is too high, we present an O(n) approximate algorithm.

Desirable Properties

We aim at finding the points where the behavior of a trajectory changes rapidly, which we call characteristic points. From a trajectory TR_i= p1p2p3…..pj…..pleniwe determine a set of characteristic points $\{p_{c1}, p_{c2}, p_{c3}, \cdots, p_{cpari}\}$ ($c_1< c_2< c_3< \cdots < c_{pari}$). Then, the trajectory TR_i is partitioned at every characteristic point, and each partition is represented by a line segment between two consecutive characteristic points. That is, TR_i is partitioned into a set of (pari−1) line segments$\{p_{c1}p_{c2}, p_{c2}p_{c3},\cdots, p_{cpari-1}p_{cpari}\}$. We call such a line segment a trajectory partition. Figure 6 shows an example of a trajectory and its trajectory partitions.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

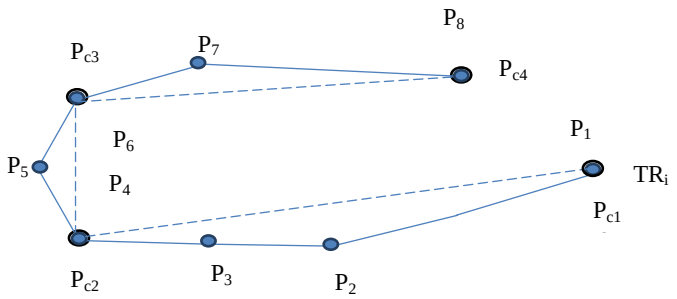

Fig.1. Example of a trajectory & its trajectory partitions

Formalization Using the MDL Principle

We propose a method of finding the optimal trade of between preciseness and conciseness. We adopt the minimum description length (MDL) principle widely used in information theory. The MDL cost consists of two components [9]: L(H) and L(D|H). Here, H means the hypothesis and D the data. The two components are informally stated as follows[9]: "L(H) is the length, in bits, of the description of the hypothesis; and L(D|H) is the length, in bits, of the description of the data when encoded with the help of the hypothesis." we formulate L(H) by Formula (1). Here, len (pcjpcj+1) denotes the length of a line segment pcjpcj+1, i.e., the Euclidean distance between pcj and pcj+1. Hence, L(H) represents the sum of the length of all trajectory partitions. On the other hand, we formulate L(D|H) by Formula (2).L (D|H) represents the sum of the difference between a trajectory and a set of its trajectory partitions.

$$L(H) = \sum_{j=1}^{par\ i\ -1} \log_2(len(\mathrm{p}_{cj}\,\mathrm{p}_{cj+1})) \qquad (1)$$

$$L\left(\frac{D}{H}\right) = \sum_{j=1}^{par\ i-1} \quad \sum_{k=cj}^{cj+1-1} \{\log_2\left(\mathrm{d}_1\left(\mathrm{p}_{cj}\,\mathrm{p}_{cj+1}, \mathrm{p}_k\,\mathrm{p}_{k+1}\right)\right) + \log_2\left(\mathrm{d}_o\left(\mathrm{p}_{cj}\,\mathrm{p}_{cj+1}, \mathrm{p}_k\,\mathrm{p}_{k+1}\right)\right)\} \qquad (2)$$

Approximate Solution

The algorithm Approximate Trajectory Partitioning shows below. We compute MDLpar and MDL_{nopar} for each point in a trajectory (lines 5~6). If MDL_{par} is greater than MDL_{nopar}, we insert the immediately previous point pcurrIndex−1 into the set CP_i of characteristic points (line 8). Then, we repeat the same procedure from that point (line 9). Otherwise, we increase the length of a candidate trajectory partition (line 11).

Algorithm Approximate Trajectory Partitioning

Input: A trajectory TR_i = p1p2p3⋯pj ⋯$plen_i$
Output: A set CP_i of characteristic points

Algorithm:

```
01:     Add p1 into the set CPi; /* the starting point */
02:     startIndex := 1, length := 1;
03:     while (startIndex + length ≤ leni) do
04:             currIndex := startIndex + length;
05:             costpar := MDLpar(pstartIndex, pcurrIndex);
06:             costnopar := MDLnopar(pstartIndex, pcurrIndex);
        /* check if partitioning at the current point makes the MDL cost larger than not partitioning */
07:             if (costpar> costnopar) then
        /* partition at the previous point */
08:                     Add pcurrIndex − 1 into the set CPi;
09:                     startIndex := currIndex−1, length := 1;
10:             else
11:             ength := length + 1;
12:     Add pleni into the set CPi;    /* the ending point */
```

2. Trajectory Clustering

In this section, we propose a trajectory clustering algorithm for the grouping phase. The procedure of relative density-based cluster algorithms finding clusters is as follow. At first, it selects any core object p from data set D, and finds the

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

core set of p, and gets the initial cluster C1. Then it expands the cluster C1 until when no new object can be added to it. When all core objects from data set D are marked as member of some clusters, and there is no new object can be added to any cluster, the algorithm is over. The expanding method used by initial cluster C1 takes a two-step procedure. First, it expands the core set of object p and gets the expanded core set of cluster C1. Second, the method used to expand cluster C1 must meet with the condition that the core objects of expanded core set are density-reachable, and its detailed expanding method can be seen in the P-code description of ExpandCluster1 function. The P-code of relative density-based cluster algorithm RDBKNN (Re1ative Density Based K-Nearest Neighbors Clustering) is described as below:

Related concepts
We describe relative density-based cluster algorithm RDBKNN with some related concepts.

Definition 1 k-distance of an object p [4] For any positive integer k and data set D, the k-distance of object p, denoted as k-distance(p), is defined as the distance d(p,o) between p and an object oϵ D such that:

(i) for at least k objects o' ∈ D \ {p} it holds that d(p,o') ≤ d(p,o), and
(ii) for at most k-1 objects o' ∈ D \ {p} it holds that d(p,o') < d(p,o)

Definition 2 k-distance neighborhood of an object p [4] Given the data set D and the k-distance of p, the k-distance neighborhood of p is defined as Nk-distance(p)(p) = { q . D\{p} | d(p, q).kdistance(p) }. It is also called the set of k-nearest neighbors of p.

Definition 3 Near Neighbor Distance of an object p w.r.t. object 0[4] Let k be a natural number. The Near Neighbor Distance of object p with respect to object o is defined as distk-distance (o) (p,o) = max { k-distance(o), d(p, 0) } ,

Definition 5 the relative density of an object p w.r,t. its Nk-distance(p) (p) neighbors Given the data set D, p ∈ D, the relative Density of p with respect to its Nk-distance(p) (p) neighbors, denoted as rdk-distance(p)(p) is defined as:

$$rd_{\text{k-distance(p)}}(p) = \frac{\sum_{o \in N\ k-distance\ (p)(P)} nnd\ k-distance\ (o)^{(o)} / nnd\ k-distance\ (p)^{(p)}}{|N\ k-distance\ (p)^{(p)}|}$$

The rdk-distance(p)(p) value reflects the difference between the near neighbors density of object p and its neighbors', when rdk-distance(p)(p) is close to 1, it illustrates that object p has a good relationship with its neighborhoods whose density are very close in data distribution, and they can be merged into a cluster very well.

Algorithm Re1ative Density Based K-Nearest Neighbors Clustering (RDBKNN)

```
RDBKNN (Set Setofpointp, int k, real α)
// α is a threshold greater than zero (α>0).
BEGIN
REPEAT
Pointp = GetCorePointp (Setofpointp, k, α);
if pointp <> null then
        Coreset1    = GetCoreSet1 (Setofpointp, pointp, k, α);
        Clustered1 = GetClusterId1 ( );
        C1 = GetInitCluster1 (Setofpointp, pointp, CoreSet1, k, clusterID1);
        ExpandCluster1 (Setofpointp, C1 , CoreSet1, k, α);
 end if
untill no more cluster can be expanded;
end RDBKNN.
The p-code of Expandcluster1 is described as follows:
Expandcluster1 (Set Setofpointp, Cluster C1, Set CoreSet1, int k, real α)
BEGIN
        Seedset1 = CoreSet1;
        while not SeedSet1.empty() DO
        Pointp = GetOutPointp1 (SeedSet1);
        NewCoreset1 = GetCoreSet1 (Setofpointp, pointp, k, α);
                for j from 1 to NewCoreset1.size do
                        object1 = NewCoreset1.get (j);
```

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

```
                    if | rd CoreSet1(object1) - 1 | < α then
                        SeedSet1 = SeedSet1 ∪ (object};
                        CoreSet1 = Coreset1 ∪ (object);
                    end if;
                end for;
            D =D ∪ nk-distance (point) (point);
            end while;
end Expandclus1.
```

3. Experimental Evaluation

In the experimental section, using the approximation algorithm we clustered & partitioned the moving object trajectory. We compare the clustering performance of our method to the widely used Euclidean and DTW distance functions.

Table1. Correspondence values and running times cycle between two sequences from our ANIMALS dataset

		Correspondence					Running time cycle in (sec)				
ω	φ	Actual time	No of tries movement				Actual time	No of tries movement			
			5	10	26	52		5	10	26	52
5	0.45	0.419	0.2045	0.42	0.323	0.433	19.905	0.0042	0.0037	0.0037	0.0042
5	0.8	0.661	0.513	0.506	0.617	0.631	19.907	0.0044	0.00271	0.00271	0.0042
6	0.45	0.489	0.376	0.358	0.806	0.233	42.452	0.0048	0.0032	0.0032	0.00491
6	0.8	8.811	0.868	0.588	0.736	0.854	42.454	0.0048	0.0043	0.0043	0.00490
7	0.45	0.603	0.320	0.453	0.673	0.485	96.245	0.00581	0.00341	0.00341	0.0042
7	0.8	0.823	0.540	0.5935	0.631	0.689	96.256	0.00581	0.0026	0.0026	0.0042

1. The Euclidean distance is only defined for sequences of the same length (and the length of our sequences varies considerably). We tried to offer the best possible comparison between every pair of sequences, by sliding the shorter of the two trajectories across the longer one and recording their minimum distance.

2. For DTW we modified the original algorithm in order to match both x and y coordinates. In both DTW and Euclidean we normalized the data before computing the distances. Our method does not need any normalization, since it computes the necessary translations.

3. For LCSS we used a randomized version with and without sampling, and for various values of ω. The time and the correct clustering's represent the average values of 15 runs of the experiment. This is necessary due to the randomized nature of our approach.

Distance Function	Time cycle in (sec)	Accurate Clusterings (out of 20) Complete Linkage
Euclidean	45.85	4
DTW	447.756	10
LCSS:		
z=10%, φ=30%	4.644	19.900
z=15%, φ=30%	10.021	19.956
z=20%, φ=30%	20.261	20
z=25%, φ=30%	38.851	20
z=30%, φ=30%	55.076	20
z=35%, φ=30%	75.309	20
z=40%, φ=30%	133.694	20
z=50%, φ=30%	455.844	20
z=100%, φ=30%	868.388	20

Table 2. Results for the vedio tracking data for various terms of sample z and φ.

IV. CONCLUSION

In this paper give the result of movement of object of trajectory animals. This paper analysis the different part of partition algorithm and giving minimum running time of trajectories point. Paper compare *N* tries of series of movement of data wild animals. Also the respective result analysis the MOTRACLUS algorithm and noisy condition. Another approach used Euclidean space and map technique for partition and groups the points.

REFERENCES

[1] P. K. Agarwal, L. Arge, and J. Erickson
[2] Efficient Similarity Search in Sequence Databases. In Proc. of the 4th FODO, pages 69–84, Oct. 1993.
[3] R. Agrawal, K. Lin, H. S. Sawhney, and K. Shim. Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases. In Proc of VLDB, pages 490–501, Sept. 1995.
[4] D. Barbara. Mobile computing and databases - a survey. IEEE TKDE, pages 108–117, Jan. 1999.
[5] D. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In Proc. of KDD Workshop, 1994.
[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In Proc of ICDT, Jerusalem, pages 217–235, 1999.
[7] B. Bollobas, G. Das, D. Gunopulos, and H. Mannila. TimeSeries Similarity Problems and Well-Separated Geometric Sets. In Proc of the 13th SCG, Nice, France, 1997.
[8] T. Bozkaya, N. Yazdani, and M. Ozsoyoglu. Matching and Indexing Sequences of Different Lengths. In Proc. of the CIKM, Las Vegas, 1997.
[9] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In Proc. of VLDB, pages 89–100, 2000.
[10] K. Chu and M. Wong. Fast Time-Series Searching with Scaling and Shifting. ACM Principles of Database Systems, pages 237–248, June 1999.
[11] G. Das, D. Gunopulos, and H. Mannila. Finding Similar Time Series. In Proc. of the First PKDD Symp., pages 88–100, 1997.
[12] R. Duda and P. Hart. Pattern Classification and Scene Analysis. John Wiley and Sons, Inc., 1973.
[13] C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo. Signature technique for similarity-based queries. In SEQUENCES 97, 1997.
[14] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proc. ACM SIGMOD, pages 163–174, May 1995.
[15] C. Faloutsos, M. Ranganathan, and I. Manolopoulos. Fast Subsequence Matching in Time Series Databases. In Proceedings of ACM SIGMOD, pages 419–429, May 1994.
[16] S. Gaffney and P. Smyth. Trajectory Clustering with Mixtures of Regression Models. In Proc. of the 5th ACM SIGKDD, San Diego, CA, pages 63–72, Aug. 1999.
[17] X. Ge and P. Smyth. Deformable markov model templates for time-series pattern matching. In Proc ACM SIGKDD, 2000.
[18] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In Proc. of 25th VLDB, pages 518–529, 1999.
[19] J. Gips, M. Betke, and P. Fleming. The Camera Mouse: Preliminary investigation of automated visual tracking for computer access. Proceedings of the Rehabilitation Engineering and Assistive Technology Society of North
[20] America 2000 Annual Conference, pages 98–100, Orlando, FL, July 2000.
[21] D. Goldin and P. Kanellakis. On Similarity Queries for TimeSeries Data. In Proceedings of CP '95, Cassis, France, Sept. 1995.
[22] J. Goldstein and R. Ramakrishnan. Contrast plots and psphere trees: Space vs. time in nearest neighbour searches. In Proc. of the VLDB, Cairo, pages 429–440, 2000.
[23] H. V . Jagadish, A. O. Mendelzon, and T. Milo. Similaritybased queries. In Proc. of the 14th ACM PODS, pages 36–45, May 1995.
[24] T. Kahveci and A. K. Singh. Variable length queries for time series data. In Proc. of IEEE ICDE, pages 273–282, 2001.
[25] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In Proc. of ACM SIGMOD, pages 151– 162, 2001.
[26] E. Keogh and M. Pazzani. Scaling up Dynamic Time Warping for Datamining Applications. In Proc. 6th Int. Conf. on Knowledge Discovery and Data Mining, Boston, MA, 2000.
[27] G. Kollios, D. Gunopulos, and V . Tsotras. On Indexing Mobile Objects. In Proc. of the 18th ACM Symp. On Principles of Database Systems (PODS), pages 261–272, June 1999.
[28] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity Search for Multidimensional Data Sequences. In Proceedings of ICDE, pages 599–608, 2000.
[29] S. Park, W. Chu, J. Yoon, and C. Hsu. Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases. In Proceedings of ICDE, pages 23–32, 2000.
[30] S. Perng, H. Wang, S. Zhang, and D. S. Parker. Landmarks: a New Model for Similarity-based Pattern Querying in Time Series Databases. In Proceedings of ICDE, pages 33–42, 2000. [30] D. Pfoser, C. Jensen, and Y. The derides. Novel Approaches in Query Processing for Moving Objects. In Proceedings of VLDB, Cairo Egypt, Sept. 2000.