A Framework for Trajectory Clustering

Elio Masciari

ICAR-CNR
Institute for the High Performance Computing of Italian National Research Council
masciari@icar.cnr.it

Abstract. The increasing availability of huge amounts of "thin" data, i.e. data pertaining to time and positions generated by different sources with a wide variety of technologies (e.g., RFID tags, GPS, GSM networks) leads to large spatio-temporal data collections. Mining such amounts of data is challenging, since the possibility of extracting useful information from this particular type of data is crucial in many application scenarios such as vehicle traffic management, hand-off in cellular networks and supply chain management. In this paper, we address the issue of clustering spatial trajectories. In the context of trajectory data, this problem is even more challenging than in classical transactional relationships, as here we deal with data (trajectories) in which the order of items is relevant. We propose a novel approach based on a suitable regioning strategy and an efficient clustering technique based on edit distance. Experiments performed on real world datasets have confirmed the efficiency and effectiveness of the proposed techniques.

1 Introduction

Trajectories are data logs pertaining time and the position of moving objects (or groups of objects) that could be generated in a wide variety of applications, such as GPS systems [5], supply chain management [16] , vessel classification by satellite images [13].

As an example of such data consider moving vehicles, where both cars and trucks leave a digital trace by the personal or vehicular mobile devices that can be collected via a wireless network infrastructures. Furthermore, mobile phones continuously signalling their locations (cell), are at each moment connected to their GSM network. When the phone sets up a call and moves through the network, there may occur the so called hand-off problem, i.e. the cell where the user is moving through does not have enough bandwidth to accomodate the call. In the same way, GPS-equipped portable devices can record their latitude-longitude position at each moment that they have a location fix, and transmit their trajectories to a collecting server. Moreover, many major retailers ask their provider to exploit RFID technology to tag pallets entering their warehouses and then moving across the supply chain. RFID readers periodically scan the tags appearing in their working area and generate data to report the object being read, the timestamp and the actual location of that object. Finally, a very peculiar type of trajectory is represented by

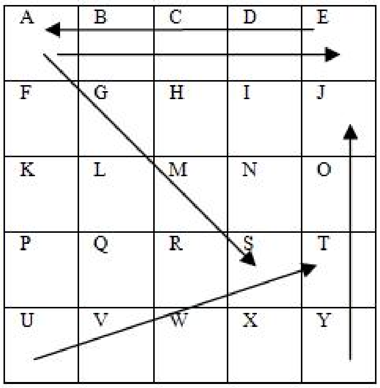

Fig. 1. A set of example trajectories

the stock market. In this case, space can be assumed as a linear sequence of points whose actual values have to be evaluated w.r.t. preceding points in the sequence in order to estimate future fluctuations.

Such a wide spectrum of pervasive and ubiquitous technologies guarantees an increasing availability of large amounts of data pertaining to individual trajectories, having a great localization precision. Therefore, due to the large amount of data generated daily by moving objects, there is a need for analyzing it efficiently in order to extract useful information.

In order to better understand the problem of trajectory clustering which is tackled by the current paper, we provide the following toy example.

Example 1. Consider the trajectories depicted in Fig. 1 and the simple regioning associated to the search space. We can represent the set of trajectories as $T = \{ABCDE, EDCBA, AGMS, UVWST, YTOJ\}$. It is easy to see that while in the transactional case the first two items would be considered the same, in our case they have to be treated separately.

Based on this representation of trajectory data, we propose a novel approach for clustering them based on suitable space partitioning. Indeed, since trajectory data carries information about the actual position and timestamp of a moving object, we can split the search space into regions with suitable granularity and represent them as symbols.

The sequence of regions defines the trajectory traveled by a given object. Note that regioning is a common assumption in trajectory data mining [13,5] and in our case it is even more suitable since our goal is to extract crucial information about trajectories' "shape", i.e. not only the length, but the direction of movement and the eventual turn made during the trajectory.. In this paper we exploit *Principal Component Analysis*(PCA) to effectively identify preferred directions for trajectories.

Once we obtain a proper regioning we can encode the trajectories as string of region symbols since we are interested in identifying the "shape" of the trajectory, while the temporal information relates only to the region traversal order as in the above example. Once the corresponding trajectory strings have been computed we use a suitable string metric based algorithm in order to evaluate the similarity of the trajectories. The most widely known string metric is *Levenshtein* Distance (also known as Edit Distance). This distance metric takes two input strings as arguments, and returns a score equivalent to the number of substitutions and deletions needed in order to transform one input string into another. The rationale for using edit distance is that it is able to capture the main behavior of different trajectories, such as different orders of regions being crossed. In the above example, trajectories $ABCDE$ and $EDCBA$ exhibit the highest dissimilarity as expected.

We performed several tests in order to assess the validity of our proposal and the results so far obtained are convincing, as will be shown in the experimental section.

Outline. In section 2 we formalize the trajectory clustering problem and our regioning strategy exploiting PCA. In section 3 we show the trajectory encoding and the edit distance based clustering strategy. In section 4 we will describe our experimental evaluation. Finally in section 5 we will draw our conclusion.

2 Problem Statement

In this paper we tackle the problem of clustering, from a large body of trajectory data. While for transactional data a tuple is a collection of features, a trajectory is an ordered set (i.e., a sequence) of timestamped items. Trajectory data are usually recorded in variety of different formats, and they can be drawn from a continuous domain. We assume a standard format for each trajectory, with the following definition:

Definition 1 (Trajectory). *Let P and T denote the set of all possible (spatial) positions and all timestamps, respectively. A trajectory is defined as a finite sequence $s_1, \cdots, s_N$, where $N \geq 1$ and each s_i is a pair (p_i, t_i) where $p_i \in P$ and $t_i \in T$.*

Throughout this paper, we assume that P is a set of discrete symbols. For continuous locations, one can partition the space into regions to map the initial locations into discrete symbols. Notice that the method chosen for the assignment of symbols to locations is totally irrelevant to our goal since we are interested in clustering the trajectory as a whole structure. The granularity of the regioning can be decided according to the application requirements. For example, for tracking trucks, the GPS data can be rounded to within 15 meters of precision and so on.

A cluster is a set of regions, and regions belonging to the same cluster are close to each other according to a suitable distance measure. The representative of the cluster is a (possibly imaginary) trajectory that summarizes the main features of the trajectories belonging to the cluster.

2.1 Defining Regions of Interest

The problem of finding a suitable partitioning for both the search space and the actual trajectory is a core problem when dealing with spatial data. Every technique proposed so far, somehow deals with regioning and several have been proposed such as partitioning of the search space in several regions of interest (*RoI*)[5] and trajectory partitioning [12,8]. In this section, we describe the application of Principal Component Analysis (*PCA*) in order to obtain a better partitioning. Indeed, *PCA* finds *preferred* directions for data being analyzed. Thus, we can exploit this information for saving both space and time. Our goal is to find interesting clusters, so it is likely that less frequently crossed regions are not significant.

2.2 Principal Component Analysis

Principal Component Analysis (PCA) finds a linear transformation l which reduces d-dimensional feature vectors to a set of h-dimensional feature vectors (where $h < d$) in such a way that the information is maximally preserved in terms of minimizing the mean squared error. The PCA also allows rolling back to d-dimensions from the h-dimensional feature vectors, with of course the introduction of some reconstruction error.

The h-column vectors define the basis vectors. The first basis vector is in the direction of maximum variance of the given feature vectors. The remaining basis vectors are mutually orthogonal and maximize the remaining variances subject to the orthogonal condition. Each basis vector represents a principal axis. These principal axes are those orthonormal ones onto which the remaining variances are maximum under projection. The orthonormal axes are given by the leading eigenvectors (i.e. those with the largest associated eigenvalues) of the measured covariance matrix. In PCA, the original feature space is characterized by these basis vectors and the number of basis vectors used for characterization is usually less than the dimensionality d of the feature space

Many tools have been implemented for computing PCA such as [19,10] – in our experiments we used a PCA algorithm based on eigenvalue decompositions.

2.3 Exploiting PCA for Regioning

Before defining our simple and effective regioning strategy we must give an intuitive interpretation of the results of PCA analysis on trajectory data. Consider the set of trajectories depicted in Fig. 2.

As shown in Fig. 3 running the PCA algorithm on the above set of trajectories identifies the preferred directions. This information can be exploited using a partition strategy that concentrates on regions along the principal directions. This allows us to focus on the preferred regions when computing clusters.

Indeed, the results of PCA is a set of eigenvalues that states the principal directions among data. Each eigenvalue defines an angle α w.r.t. the principal axes. Depending on the density of data we choose a suitable level of granularity

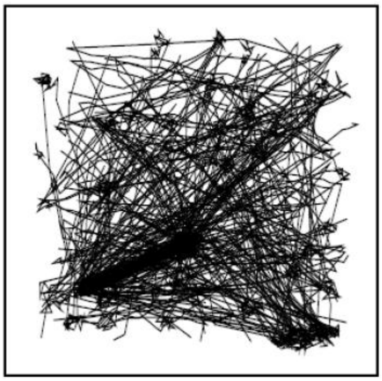

Fig. 2. A set of trajectories

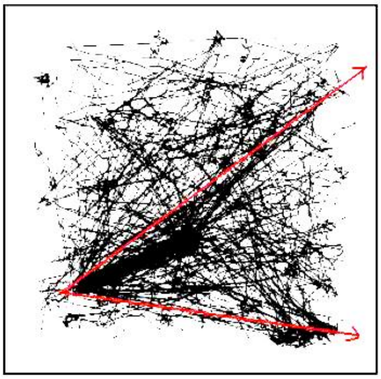

Fig. 3. Principal eigenvalues identified by PCA

that defines the size d of each region (we assume square regions). The size of each region is defined according to the domain being analyzed, for example for cellular nets it could be some meters while for truck movements it could be hundred of meters, while for hurricane control it will be in the order of kilometers. In order to define regions of interest we divide the search space into square regions along the directions set by the eigenvalues so far obtained by PCA and having size d. It will happen that this regioning will also consider regions not parallel to the principal axes, but will not cover the whole search space. Indeed, the "white" regions, i.e. regions not on the principal directions, could be considered as not being interesting regions and thus requiring no further investigation. This strategy is quite effective in practice since we note that infrequently visited regions will not affect clustering results. Thus, pruning the search space will increase the efficiency and effectiveness of the mining step.

2.4 Trajectory Regioning

Once the set of regions $R = \{R_1, R_2, \cdots, R_m\}$ is obtained as explained in the previous sections, we assign to each region a symbol from a given alphabet Σ.

A trajectory Tr_i is a sequence of multidimensional points $Tr_i = p_1, p_2, \cdots, p_n$ where n is the trajectory length. Given a set of trajectories T, we define $T' = \mathit{encode}(T)$ the set of encoded trajectories. An encoded trajectory T'_j is a sequence of regions $T'_j = R_{j,1}, R_{j,2}, \cdots, R_{j,n}$. The encoding is performed by simply substituting each point p_i with the region R_i it belongs to as shown in Fig. 4.

```
Run PCA(R)
For Each trajectory $Tr_i$
1:  For each point $p_i \in Tr_i$
2:  find $R_{j,i}$ containing $p_i$
3:  append $R_{j,i}$ to $T'_j$
```

Fig. 4. PCA regioning pseudo code

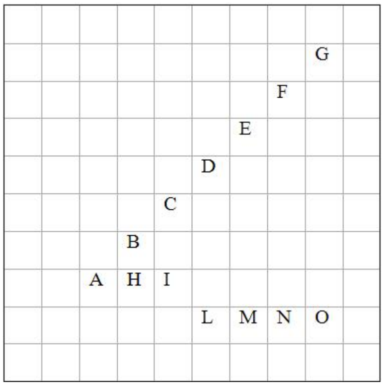

Fig. 5. Regions defined by PCA

Fig. 5 shows a regioning obtained for the set of trajectories from Fig. 3. The encoding phase allows us to obtain a set of trajectories that can be mined using the algorithms that will be explained in next sections.

3 Exploiting Edit Distance for Clustering Trajectories

In order to efficiently identify trajectory clusters, we need to define a measure of similarity between the two sequences. Intuitively, two trajectories are said to have a similar structure if they correspond in the regions crossed and in the order the regions are traversed. Indeed we would like to quantify the similarity between the two trajectories, also emphasizing the differences that are more relevant. For instance, we consider as similar two trajectories that exhibit the same features with different delays since this could be due to a simple traffic problem as will be clear in the next section.

Given two strings their edit distance is defined as the minimum number of operations needed to transform one string into the other. The allowed operations are insertion, deletion, or substitution of a single character, note the the transposition in our case could be seen as the composition of a deletion and an insertion. The string metric so far defined is called *Levenshtein distance.*

Given two encoded trajectories t'_a, t'_b we define $ED(t'_a, t'_b)$ their edit distance. We implemented the algorithm in [14]. In particular we use the following cost assignment:

– copying a region from t'_a over to t'_b (cost 0);
– deleting a region in t'_a (cost 1);
– insert a region in t'_b (cost 1);
– substituting one region for another (cost 1).

Thus $ED(t'_a, t'_b)$ is the minimum among the following:

1. $ED(i-1, j-1) + d(t'_a(i), t'_b(j))$ (substitution or copy);
2. $ED(i-1, j) + 1$ (insert);
3. $ED(i, j-1) + 1$ (delete).

Note that $d(i, j)$ is a function defined as $d(i, j) = 0$ if $t'_a(i) = t'_b(j)$, 1 otherwise, where i and j are symbol positions in a given trajectory.

4 Experimental Results

In this section, we present some experiments we performed to assess the effectiveness of the proposed approach in clustering trajectories. To this purpose, a collection of tests is performed, and in each test some relevant groups of homogeneous trajectories (*trajectory classes*) are considered. The direct result of each test is a similarity matrix representing the degree of similarity for each pair of trajectories in the data set. The evaluation of the results relies on some *a priori* knowledge about the trajectory being used. We performed several experiments on a wide variety of real datasets. We analyzed the following data:

– *School Bus*: a dataset consisting of 145 trajectories of 2 school buses collecting (and delivering) students around Athens metropolitan area in Greece for 108 distinct days[1];
– *Trucks*: a dataset consisting of 276 trajectories of 50 trucks delivering concrete to several construction places around Athens metropolitan area in Greece for 33 distinct days[2];

[1] Available at http://www.rtreeportal.org
[2] Available at http://www.rtreeportal.org

– *Animals*: a dataset containing the major habitat variables derived for radio-telemetry studies of elk, mule deer, and cattle at the Starkey Experimental Forest and Range in northeastern Oregon[3].

The similarity matrix enables simple quantitative analyses, aimed at evaluating the resulting intra-class similarities (i.e., the average of the values computed inside each class), and to compare them with the inter-class similarities (i.e., the similarity computed by considering only documents belonging to different classes). To this purpose, values inside the matrix can be aggregated according to the classes of membership of the related elements: given a set of trajectories belonging to n prior classes, a similarity matrix S about these trajectories can be summarized by a $n \times n$ matrix CS, where the generic element $CS(i, j)$ represents the average similarity between class i and class j:

$$CS(i,j) = \begin{cases} \frac{\sum_{x,y \in C_i, x \neq y} ED(x,y)}{|C_i| \times (|C_i|-1)} & \text{iff } i = j \\ \frac{\sum_{x \in C_i, y \in C_j} ED(x,y)}{|C_i| \times |C_j|} & \text{otherwise} \end{cases}$$

The higher the values on the diagonal of the corresponding CS matrix are w.r.t. those outside the diagonal, the greater the ability of the similarity measure to separate different classes. In the following results we report a similarity matrix for each dataset being considered, as it easy to see it has proven that the technique is quite effective for clustering the datasets being considered.

4.1 School Bus

For this dataset our prior knowledge is the set of trajectories related to the two school buses. We present the results using two classes but we point out that our technique is able to further refine the cluster assignment by identifying the microclusters represented by common subtrajectories. The results are shown in Table 1. It is clear to see from the results that there is a crisp difference between the two clusters.

Table 1. Average similarities for the School Bus dataset

	Bus 1	Bus 2
Bus 1	0.9990	0.7528
Bus 2	0.7528	1

4.2 Trucks

In this case we considered as a class assignment the different trajectories reaching the area where concrete were delivered. In this case there were six main classes as it is shown in Table 2.

[3] Available at http://www.fs.fed.us/pnw/starkey/data/tables/index.shtml

Table 2. Average similarities for the Trucks dataset

	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
Site 1	0.9981	0.7608	0.7610	0.8125	0.8145	0.8275
Site 2	0.7608	0.9909	0.7176	0.7494	0.7150	0.7968
Site 3	0.7610	0.7176	0.9898	0.6054	0.7994	0.7021
Site 4	0.8125	0.7494	0.6054	0.9994	0.7500	0.7515
Site 5	0.8145	0.7150	0.7994	0.7500	0.9970	0.7994
Site 6	0.8275	0.7968	0.7021	0.7515	0.7994	0.9894

4.3 Animals

In this case we considered as a class assignment the different trajectories traversed by elk, mule deer, and cattle. Thus, in this case there were three main classes as shown in Table 3.

Table 3. Average similarities for the Animals dataset

	elk	mule deer	cattle
elk	1.000	0.6639	0.6668
mule deer	0.6639	0.9993	0.7005
cattle	0.6668	0.7005	0.9995

5 Conclusion

In this paper we addressed the problem of detecting clusters in trajectory data. The technique we have proposed is mainly based on the idea of representing a trajectory as string. Thereby, the similarity between two trajectories can be computed by exploiting the edit distance between the associated strings. Experimental results showed the effectiveness of our approach. The current work is subject to further extensions, that we plan to address in future work such as exploiting more information about the regions obtained with PCA and defining a regioning strategy that allows more complex regions. Moreover we would like to investigate a more refined edit distance algorithm in order to better capture the main features of the trajectories being analyzed, i.e. to take into account the Euclidean distances between regions.

References

1. Agrawal, J., Diao, Y., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: SIGMOD Conference, pp. 147–160 (2008)
2. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)

3. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM 7(3), 171–176 (1964)
4. Agrawal, R., et al.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD (1998)
5. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: KDD, pp. 330–339 (2007)
6. Goldin, D., Kanellakis, P.: On similarity queries for time-series data: Constraint specification and implementation. In: Montanari, U., Rossi, F. (eds.) CP 1995. LNCS, vol. 976, pp. 137–153. Springer, Heidelberg (1995)
7. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)
8. Jae-Gil, L., Jiawei, H., Kyu-Young, W.: Trajectory clustering: a partition-and-group framework. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 593–604. ACM, New York (2007)
9. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of convoys in trajectory databases. PVLDB 1(1), 1068–1080 (2008)
10. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics (2002)
11. Kéri, G., Kisvölcsey, Á.: On computing the hamming distance. Acta Cybernetica 16(3), 443–449 (2004)
12. Lee, J.-G., Han, J., Li, X.: Trajectory outlier detection: A partition-and-detect framework. In: ICDE, pp. 140–149 (2008)
13. Lee, J.-G., Han, J., Li, X., Gonzalez, H.: *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. PVLDB 1(1), 1081–1094 (2008)
14. Levenshtein: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966)
15. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: KDD, pp. 617–622 (2004)
16. Liu, Y., Chen, L., Pei, J., Chen, Q., Zhao, Y.: Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. In: PerCom., pp. 37–46 (2007)
17. Rafiei, D., Mendelzon, A.: Efficient retrieval of similar time series. In: Procs. 5th Int. Conf. of Foundations of Data Organization (FODO 1998) (1998)
18. Sadri, R., Zaniolo, C., Zarkesh, A.M., Adibi, J.: Expressing and optimizing sequence queries in database systems. ACM Trans. Database Syst. 29(2), 282–318 (2004)
19. Sharma, A., Paliwal, K.K.: Fast principal component analysis using fixed-point algorithm. Pattern Recognition Letters 28(10), 1151–1155 (2007)
20. Yang, J., Hu, M.: Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 664–681. Springer, Heidelberg (2006)