Space- and Time- Invariant Trajectory Clustering via Deep Representation Learning

Di Yao[1,3], Chao Zhang[2], Zhihua Zhu[1,3], Jianhui Huang[1,3], Jingping Bi[1,3]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[3]University of Chinese Academy of Sciences, Beijing, China
Email: [1,3]{yaodi, zhuzhihua, huangjianhui, bjp}@ict.ac.cn, [2]czhang82@illinois.edu

***Abstract*—Trajectory clustering, which aims at discovering groups of similar trajectories, has long been considered as a corner stone task for revealing movement patterns as well as facilitating higher-level applications like location prediction. While a plethora of trajectory clustering techniques have been proposed, they often rely on spatiotemporal similarity measures (e.g., Dynamic Time Warping) that are not space- and time-invariant. As a result, they cannot detect trajectory clusters where the within-cluster similarity occur in different regions and time periods. In this paper, we revisit the trajectory clustering problem by learning quality low-dimensional representations of the trajectories. We first use a sliding window to extract a set of moving behavior features that capture space- and time-invariant characteristics of the trajectories. With the feature extraction module, we transform each trajectory into a feature sequence to describe object movements and further employ a sequence to sequence auto-encoder to learn fixed-length deep representations. The learnt representations robustly encode the movement characteristics of the objects and thus lead to space- and time- invariant clusters. We evaluate the proposed method on both synthetic data and real data sets and observe significant performance improvements over existing methods.**

I. Introduction

Owing to the rapid growth of GPS-equipped devices and location-based services, enormous amounts of spatial trajectory data are being collected in different scenarios. Among various trajectory analysis tasks, trajectory clustering — which aim at discovering groups of similar trajectories — has been recognized as one of the most important. Discovering trajectory clusters can not only reveal the latent characteristics of the moving objects, but also support a wide spectrum of high-level applications, such as travel intention inference, location prediction and anomaly detection[1].

A plethora of trajectory clustering techniques have been proposed [2], most of which use certain measures to quantify trajectory similarities and then apply classic clustering algorithms (e.g., K-means, DBSCAN, spectral clustering). Popular trajectory similarity measures[3] include DTW(Dynamic Time Warping), EDR(Edit Distance on Real sequence) and $LCSS$(Longest Common Subsequences). Although these measures can group trajectories that are similar in a fixed geographical region and time period, many practical applications involve trajectories that distribute in different regions with different lengths and sampling rates. In such applications, one is often required to find clusters where the within-cluster similarity occur in different time and space. For example, the taxis in traffic jams can have similar moving behaviors, but the traffic jams usually occur in different areas in the city with different durations. Such spatio-temporal shifts [4] are common in many types of moving objects and render current trajectory clustering algorithms ineffective.

In this work, we revisit the trajectory clustering problem by developing a method that can detect space- and time- invariant trajectory clusters. Our method is inspired by the recent success of recurrent neural networks (RNNs) for handling sequential data in Speech Recognition and Neural Language Processing. Given the input trajectories, our goal is to convert each trajectory into a fixed-length representation that well encodes the object's moving behaviors. Once the high-quality trajectory representations are learnt, one can easily apply any classic clustering algorithms according to practical needs. Nevertheless, it is nontrivial to directly apply RNN to the input trajectories to obtain quality representations because of the varying qualities and sampling frequencies of the given trajectories. We find that a naive strategy that considers each trajectory as a sequence of three-dimensional records (time, latitude, longitude) leads to dramatically oscillating parameters and non-convergence in the optimization process of RNNs.

In light of the above issue, we first extract a set of movement features for each trajectory. Our feature extraction module is based on a fixed-length sliding window, which scans through the input trajectory and extract space- and time-invariant features of the trajectories. After feature extraction, we convert each trajectory into a feature sequence to describe the movements of the object and employ a sequence to sequence(Seq2Seq) auto-encoder to learn fixed-length deep representations of the objects. The learnt low-dimensional representations robustly encode different movement characteristics of the objects and thus lead to high-quality clusters.

In summary, we make the following contributions:

- We study the problem of detecting space- and time-invariant trajectory clusters. Such a task differs from previous works in that it can group trajectories collected in different regions with varying lengths and sampling rates; and it can support a wide spectrum of trajectory analysis applications in practice.
- We employ a sliding-window-based approach to extract a set of robust movement features, and then apply se-

quence to sequence auto-encoders to learn fixed-length representations of the trajectories. To the best of our knowledge, this is the first study that leverages recurrent neural networks(LSTM) for the trajectory clustering task.
- We evaluate our method on both synthetic and real-life data. We find that our method can generate high-quality clusters on both data sets and largely outperforms existing methods quantitatively.

The rest of this paper is organized as follows. In Section II, we review related work on trajectory pattern discovery. We give a general overview of our framework in Section III and then detail the main steps in Section IV. We empirically evaluate the proposed method in Section V and finally conclude in Section VI.

II. Related Works

In this section, we briefly review the existing approaches for trajectory clustering and sequence to sequence auto-encoder.

A. Trajectory Clustering

Most existing trajectory clustering approaches [2] apply distance-based or density-based clustering algorithms by introducing similarity measures tailored for trajectory data[24], such as DTW (Dynamic Time Warping), EDR (Edit Distance on Real sequence) and $LCSS$ (Longest Common Subsequences). Meanwhile, Lee *et al.* [5] proposed a framework which first partitioned the whole trajectories and then grouped sub-trajectories using density-based clustering method. Tang *et al.* [6] presented a travel behavior clustering algorithm, which combined sampling techniques with density-based clustering to deal with the noise in trajectory data. Li *et al.* [7] proposed an incremental framework to support online incremental clustering. Besse *et al.* [2] performed distance-based trajectory clustering by introducing a new distance measurement. While the aforementioned methods can cluster trajectories that are similar in a fixed region and period, they are inapplicable for discovering similar moving behavior trajectory clusters in different space and time. Kohonen *et al.*[9][8] developed SOM(Self-Organizing Maps) and LVQ(Learning Vector Quantization) and it were employed in trajectory analysis and clustering.

A number of methods have been proposed for pattern-based trajectory clustering. Hung *et al.* [4] proposed a trajectory pattern mining framework that extracted frequent travel patterns and then trajectory routes. Higgs *et al.* [10] proposed a framework for the segmentation and clustering of car-following trajectories based on state-action variables. Chao Zhang [11] used the hidden Markov Model to extract mobility features to find groups of users that have similar mobility patterns. Liu *et al.* [12] introduced a speed-based clustering method to detect taxi charging fraud behavior. Different from our work that detects general trajectory clusters, this group of works perform clustering based on a special moving pattern such as car-following pattern and taxi fraud pattern.

B. Sequence to Sequence Auto-encoder

Sequence to sequence auto-encoder was first proposed by Sutskever *et al.* [14] for machine translation. Dai *et al.* [15] introduced a sequence to sequence auto-encoder and used it as a "pretraining" algorithm for a later supervised sequence learning. Recent research has also demonstrated the usefulness of sequence to sequence auto-encoders for generating a fixed-length representations for videos and sentences. Sepcifically, Chung *et al.* [16] employed it to generate audio vector; Nitish Srivastava [17] used multilayer Long Short Term Memory (LSTM) networks to learn representations of video sequences; Hamid Palangi[18] generated a deep sentence embedding for information retrieval. However, we are not aware of any previous works that apply auto-encoders to trajectory data. In addition, as aforementioned, directly applying auto-encoders on trajectory data is non-trivial because of the varying sampling frequencies and the noise between continuous records.

III. General Framework

In this section, we first formally present the problem formulation of this work in Subsection A. Then, we give an overview of our framework.

A. Problem Formulation

We denote the objects set as $O = \{o_1, o_2, ..., o_L\}$. For a specific object o, a sequence of GPS records which contains all the information for a object is given $S_o = (x_1, x_2, ..., x_M)$. S is a time series and each of x is a tuple which consists of (t_x, l_x, a_x, o_x). t_x is the timestamp; l_x is a two-dimensional vector(longitude and latitude) representing the object's location; a_x is a attributes vector collected by other sensor(If object is a car, a_x may be include the speed, turning rate, fuel consumption, etc); o_x is the object ID. S can be segmented into trajectory sequence by next DEFINATION and denoted as $TR_o = (TR_1, TR_2, ..., TR_n)$.

DEFINITION: Given $S_o = (x_1, x_2, ..., x_M)$ and a time gap threshold $\Delta t > 0$ a subsequence $S_o^T = (x_i, x_i + 1, ..., x_i + k)$ is a trajectory of S_o if S_o^T satisfies: (1). $\forall 1 < j \leq k, t_{x_j} - t_{x_{j-1}} \leq \Delta t$ and (2). there is no subsequence in S_o that contain S_o^T also satisfy condition (1).

Figure 1 depicts a simple object trajectory generation process. With $S_o = (x_1, x_2, x_3, x_4, x_5, x_6, x_7))$ and $\Delta t = 3hour$, the sequence is segmented into three trajectories. We denotes the trajectory sequence of object o as $TR_o = (TR_1, TR_2, TR_3)$.

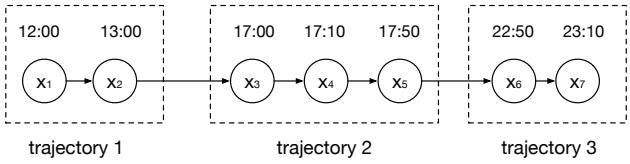


Fig. 1. Trajectory Partition.

By combining the trajectories of all the objects, we obtain a trajectory set $\mathscr{T} = \{TR_1, TR_2, ..., TR_N\}$.

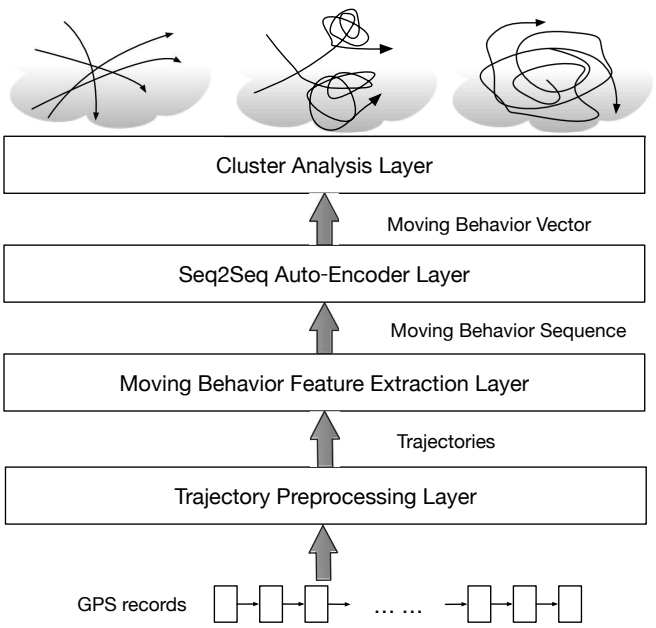


Fig. 2. The general framework for moving behavior trajectory clustering.

Our goal is to generate space- and time- invariant trajectory clusters of $\mathscr{T}$ based on the object movement patterns. Specifically, based on objects' movement patterns, we need to generate a set of clusters $\mathscr{C} = \{C_1, C_2, ..., C_K\}$. In each cluster, the similarity shared by the member trajectories may exhibit in different geographical regions and also appear in different parts of the member trajectories.

B. Overview of The Framework

We present the proposed framework, aiming to find the space- and time- invariant trajectory clusters with similar moving behavior. As depicted in Figure 2, the framework is an unsupervised approach. It consists of four layers detailed as followed.

- **Trajectory Preprocessing Layer:** The input of this layer is the GPS records sequence of the moving object. The sequence is noisy and the temporal gaps between some record pairs can be very large. In this layer, we remove the low-quality GPS records and cut apart the sequence into trajectories with temporal continuity which is illustrated in Fig.1.
- **Moving Behavior Feature Extraction Layer:** In this layer, all the trajectories are processed with a moving behavior feature extraction algorithm. Based on a sliding windows, we transform the trajectory into a feature sequence by computing the statics of the moving behaviors.
- **Seq2Seq Auto-Encoder Layer:** As the trajectories have different lengths, we use a sequence to sequence auto-encoder to embed each feature sequence to a fixed-length vector. This vector represents movement pattern of the trajectory.
- **Cluster Analysis Layer:** Finally, we choose a classic clustering algorithm based on the practical needs and cluster the learnt representations into clusters. Just as shown in Figure 1., each cluster represents a set of trajectories which sharing the similar moving behavior.

IV. Methodology

In this section, we elaborate two layers which are critical to our framework. Subsection A specifies the behavior feature extraction algorithm. Subsection B describes the structure of sequence to sequence auto-encoder.

A. Moving Behavior Feature Extraction

The key point of the behavior feature extraction is that we utilize sliding window to select the records and employ the behavior feature extraction algorithm for each window. As shown in Figure 3, with a sliding window, we can get the the space- and time- invariant features and generate a feature sequence to describe the moving behavior of the trajectory.

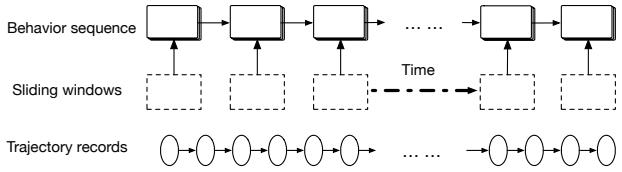


Fig. 3. Moving behavior extraction.

Considering the sampling frequency of a trajectory can be various, we utilize sliding window method in this procedure. Now we describe the detailed feature extraction process in each window. Let L_p and $offset_p$ denote the width and offset of the sliding window, respectively. While classic methods often choose $offset_p = L_p$, we find that a finer granularity of $offset_p = 1/2 \times L_p$ can effectively decrease the loss of useful information and lead to better performance. In this way, each record in a trajectory is assigned into two windows, and all the behavior changes are captured. Since the density of records is imbalanced, some dummy windows that have no records are also introduced, such as W_6 in Figure 4.

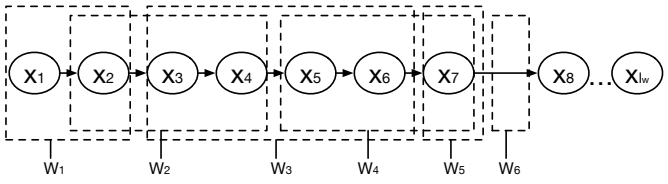


Fig. 4. Sliding time windows generation.

In the moving behavior feature extraction procedure, we extract the moving behavior features that capture space- and time- invariant characteristics in each window. The moving behavior change can be reflected by the differences of attributes between two consecutive records. Let us consider a window

with R records. The records in this window are denoted as $W = (x_1, x_2, ..., x_R)$. Assume the attributes in each record consist of speed and rate of turn(ROT). ROT reflects the moving direction change of object. For the two consecutive records x_i and x_{i-1}. The attributes reflecting the moving behavior include: time interval $\Delta t_i = t_{x_i} - t_{x_{i-1}}$, change of position $\Delta l_i = l_{x_i} - l_{x_{i-1}}$, change of speed $\Delta s_i = s_{x_i} - s_{x_{i-1}}$ and change of ROT $\Delta r_i = r_{x_i} - r_{x_{i-1}}$, where i range from 2 to R. In this way, a window with R records has $R-1$ the moving behavior attributes $(\Delta l, \Delta s, \Delta r)$.

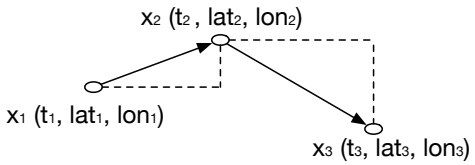


Fig. 5. Attributes completely.

Even if these are no attributes in the row record, the speed and ROT also can still be calculated according to location information. As shown in Figure 5, there is a trajectory with T records $TR = (x_1, x_2 ... x_T)$. We only has the timestamp and location coordinate in each record and denote as (t, lat, lon). For the first record of the trajectory x_1, we set $s_{x_1} = 0$ and $r_{x_1} = 0$. Then we can calculate the speed and ROT of each record by:

$$s_{x_i} = \frac{\sqrt{(lat_{x_i} - lat_{x_{i-1}})^2 + (lon_{x_i} - lon_{x_{i-1}})^2}}{t_{x_i} - t_{x_{i-1}}} x \quad (1)$$

and

$$r_{x_i} = \arctan \frac{lon_{x_i} - lon_{x_{i-1}}}{lat_{x_i} - lat_{x_{i-1}}} \quad (2)$$

where i range from 2 to T. After this procedure, the speed and ROT attributes can be derived for each trajectory.

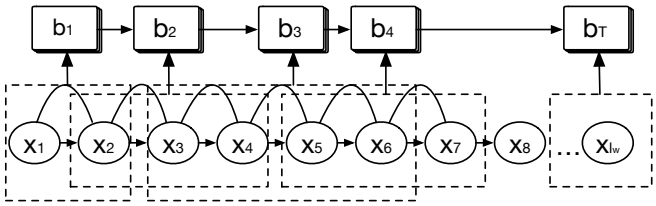


Fig. 6. The generation of moving behavior sequence.

If $R \geq 1$, for each i from 1 to R, we compute Δt_i, Δl_i, Δs_i and Δr_i. We further compute the change rate of these features $f_i = (f_{\Delta l_i}, f_{\Delta s_i}, f_{\Delta r_i})$ in which $f_{\Delta l_i} = \Delta l_i / \Delta t_i$, $f_{\Delta s_i} = \Delta s_i$ and $f_{\Delta r_i} = \Delta r_i$. For two consecutive records, $f_{\Delta l_i}$ stands for the average speed, $f_{\Delta s_i}$ stands for the change of speeds and $f_{\Delta r_i}$ stands for the change of ROTs. After computing these features in each pair, we get a feature set $f = \{f_1, f_2, ..., f_R\}$. This is a simple situation that the attributes of record only contain speed and ROT. We use the statistic of f to generate the features in the sliding window. Here, six statistics $\{mean, max, 75\%quantile, 50\%quantile, 25\%quantile, min\}$ are selected to represent the moving behavior in this window. In summary, the moving behavior features of this window b has $3 \times 6 = 18$ dimensions that consist of

$$\{f_{\Delta l}, f_{\Delta s}, f_{\Delta r}\} \times$$
$$\{mean, max, 75\%quantile, 50\%quantile, 25\%quantile, min\}$$

If $R = 0$, we skip this window. Algorithm 1 shows the generation procedure of moving behavior feature sequence.

Algorithm 1 Behavior Feature Extraction Algorithm

Input:
GPS records for a trajectory TR
Output:
The behavior sequence of trajectory TR, B_{TR}

```
Initialize B_TR = []
windows = sliding_windows(TR)
for each window W in windows do
    if len(W.records) ≥ 1 then
        Initialize F_W = []
        for each record r_i in W.records do
            r_{i-1} = find_pre(r_i)
            F_i = compute_features(r_i, r_{i-1})
            F_W.add(F_i)
        end for
        B_W = generate_behavior(F_W)
        B_TR.add(B_W)
    end if
end for
return B_TR
```

For each trajectory in $\mathscr{T}$, we generate the moving behavior sequences of its trajectories. Then, we put these sequences in a set and denote as $BS = \{B_{TR_1}, B_{TR_2}, ..., B_{TR_N}\}$. Then we normalize each feature to prepare for the next sequence to sequence auto-encoder layer.

B. LSTM Seq2Seq Auto-encoder

In this section, we describe a model that uses LSTM (Long Short Term Memory) to reconstruct the moving behavior sequence and generate a fixed-length deep representation of the trajectory[13].

RNNs are neural networks whose hidden units form a directed cycle and it is suitable for variable length inputs or outputs as a main toll for handling sequential data. For a given behavior sequence $B_{TR_i} = (b_1, b_2, ..., b_T)$ where $i \in [1, N]$, RNNs update its hidden state h_t according to the current input b_t and the previous h_{t-1}. The hidden state h_t acts as an internal memory at time t that enables the network to capture dynamic temporal information. At time t the RNN is updated by

$$h_t = f(h_{t-1}, b_t) \quad (3)$$

where f is the activation function. However, traditional hidden unit with $sigmoid$ or $tanh$ activation function does not seem

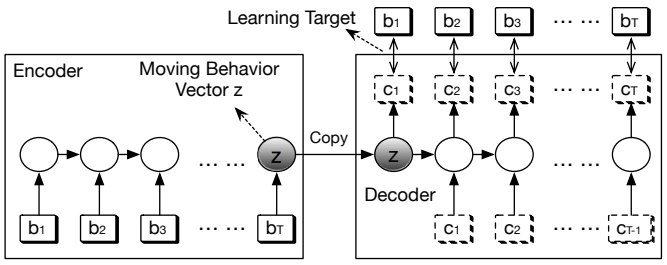


Fig. 7. Architecture of sequence to sequence auto-encoder.

to learn long-term dependencies in practice[20]. LSTM acting as a special hidden unit is proposed to overcome this shortage. RNNs equipped with LSTM has been found extremely successful in a number of application[14][15][16][19][17][18].

LSTM auto-encoder model is composed of two RNNs - the encoder LSTM as shown in the left part of Figure 7 and the decoder LSTM is illustrated in the right part of Figure 7. The input of the model is a behavior sequence B_{TR_i}. Encoder LSTM reads the input sequence sequentially and the hidden state h_t is updated accordingly. The encoder LSTM is updated by:

$$h_t = f_{LSTM}(h_{t-1}, b_t) \tag{4}$$

After the last b_T is processed, the hidden state h_T is interpreted as the learned moving behavior vector of the whole input sequence. Then, the decoder first generates the output c_1 by taking h_T as the initialize hidden state of decoder LSTM. Then take c_1 as input to generate $(c_2, c_3, ..., c_T)$. D is the dimension of the behavior features in each sliding window. The decoder LSTM is updated by:

$$h^d{}_t = f_{LSTM}(h^d{}_{t-1}, c_{t-1}, h_T) \tag{5}$$

Based on the principles of auto-encoder[15], the target of the output sequence $C_{TR_i} = (c_1, c_2, ..., c_T)$ is the input sequence $B_{TR_i} = (b_1, b_2, ..., b_T)$. In other words, the encoder LSTM and decoder LSTM are trained together by minimizing the reconstruction error, measured by the general mean squared error $\sum_{t=1}^{T} \parallel b_t - c_t \parallel^2$. The input sequence is taken as the learning target, the training process does not need any labeled data. The fixed-length moving behavior vector z will be a meaningful representation for the input behavior sequence B_{TR_i}, because the hole input sequence can be reconstructed from z by the LSTM decoder.

After this procedure, we get the moving behavior vector set $Z = \{z_{TR_1}, z_{TR_2}, ..., z_{TR_N}\}$. Then, we feed them in a classic clustering algorithm, such as K-means which is used in our experiment, and obtain the clusters.

V. Experiment

In this section, the performance of our proposed approach was tested on both synthetic dataset and real-life AIS vessel motion data in China. We first introduce the dataset of the experiments, and describe the compared methods. Then, we present the experiment results in these two datasets.

A. Dataset & Compared Methods

1) ***Dataset and Settings****:* We use both synthetic and real datasets to test the effectiveness of the framework. The results on the synthetic data quantitatively demonstrate our framework has much better performance than existing trajectory clustering methods; while the experiments on the vessel motion dataset indicate that the proposed approach find meaningful trajectories clusters. Moreover, the moving behavior vector of trajectory can be used in clustering different types of vessel.

For the synthetic dataset, we simulated 3000 trajectories including three kinds of movement patterns $\{Straight, Circling, Bending\}$. Each pattern has 1000 trajectories. The sampling frequency and time length of each trajectory were generated randomly from 2500 seconds to 5000 seconds. We first generated the time interval, which was random sampled from $Int(20, 50)$. Then, locations were generated accordingly. After that, we completed the attributes of location by using equation (1) and (2). Even through our approach can capture similar moving behavior patterns in different space-time, in order to compare with other trajectory clustering methods, we generated all the trajectories with start time equals to 0s and start location equals to $(0, 0)$. In addition, we added Gaussian noise in the location generation process. Part of the synthetic dataset is shown in Figure 8.

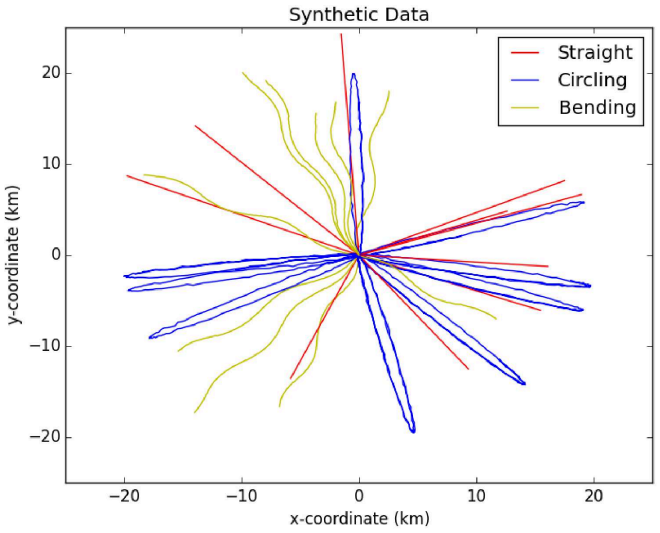


Fig. 8. Part of the synthetic data which consist of 10 straight trajectories, 10 circle trajectories and 10 bending trajectories.

The real dataset corresponds to 200 vessels in China, containing 50 cargo ships, 50 fishing ships, 50 oil ships and 50 passenger ships. The vessel motion data is collected by AIS(Automatic Identification System). AIS[21] is one of the most important ways for maritime domain awareness. AIS messages can be divided into dynamic messages and static messages. Dynamic message reports the dynamic situation of the vessel which includes the time, position (longitude, latitude), COG(course over ground), SOG(speed over ground) and heading. Static messages includes type, name and size.

The record time of these vessel ranges from 2016.5 to 2016.6. The records in this dataset are up to 5,924,142. After the trajectory partition, we generated 4700 trajectories.

We implemented the framework with Python and TensorFlow. All the experiments were performed on a server with Intel Xeon CPU 2.10GHz. The data and code are publicly available.

*2) **Compared Methods**:* We compare our method with four well-known trajectory clustering methods that employ differnt measures, including LCSS, DTW, EDR and Hausdorff distance. All the distance functions can handle trajectories with different lengths. LCSS, DTW and EDR are warping-based distance[3] function which aims to solve local time shifting problems. They enable matching locations from different trajectories with different indexes. Besides, they also find an optimal alignment between two trajectories, according to a given cost ε between the matched location. In contrast, Hausdorff distance is shape-based distance. Firstly, it defines the distance from a point to trajectory. Secondly, it computes the distance of the two trajectories. Based on each measure, we choose K-medoids (KM) as the clustering algorithm. On the synthetic data, as the number of moving behavior patterns are known, we set the number of clusters to 3. In the experiments of the real-life data, we tune the number of clusters and analyze the meaning of each cluster. For the vessel type clustering task, we evaluate the cluster performance in each vessel type.

We establish a measurement of the cluster results with the precision, recall and accuracy[25]. For each baseline method, we first determine the main cluster allocation of the movement patterns by counting the trajectory numbers of clusters. Then, for each movement pattern, we measure the result by precision and recall. The precision and recall are computed as: Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$, respectively. TP(Ture Positive), stands for the number of trajectories with this movement pattern and were allocated in the main cluster. Furtherly, FP(False Positive) stands for the number of trajectories in other patterns and were allocated in the main cluster while FN(False Negative) stands for the number of trajectories with this pattern but were not allocated in the main cluster. Finally, we measure the accuracy of baseline method. The accuracy is computed as follows: Accuracy $=$ **Sum of All TPs** / **Number of Trajectories**.

B. Results on Synthetic Dataset

In this experiment, we set sliding window size as 600s and the offset of the window as 300s. For the sequence to sequence auto-encoder procedure, we set learning rate to 0.0001 and set the iteration num to 1000. We also set the hidden state number for the LSTM cell to 100. We choose K-means algorithm to generate the trajectory clusters. Since EDR and $LCSS$ need a threshold of distance to determine whether two records are matching. After tuning, we set the distance equal to 100m. The cluster evaluation result is shown in Table. 1.

[0]https://github.com/yaodi833/trajectory2vec.git

TABLE I
TABLE OF TRAJECTORY CLUSTERING RESULTS

	Straight	Circling	Bending	Accuacy
EDR + KM	0.55/0.62	0.60/0.67	0.79/0.59	61.83%
LCSS + KM	0.54/0.46	0.56/0.75	0.50/0.40	53.6%
DTW + KM	0.45/0.54	0.51/0.45	0.49/0.44	47.73%
Hausdorff + KM	0.34/0.37	0.33/0.34	0.40/0.35	35.5%
Our Method	0.88/0.97	0.90/0.87	0.85/0.78	87.5%

This table shows the cluster result of synthetic dataset. The two numbers in each cell stand for **Precision** / **Recall** accordingly.

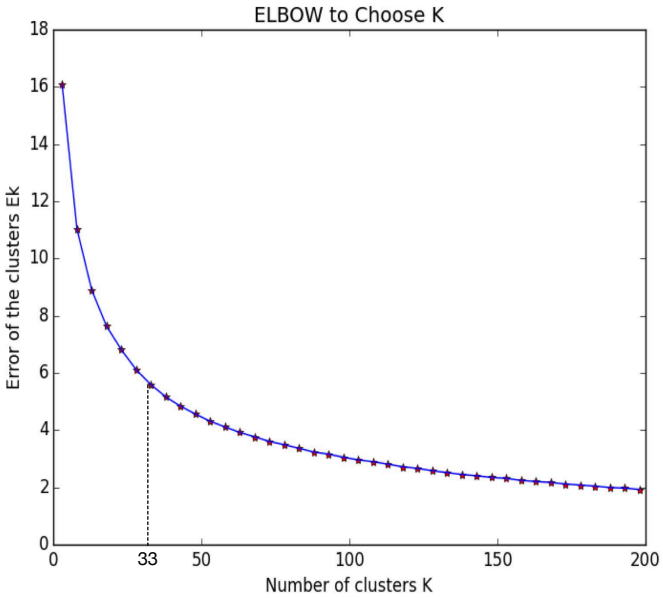


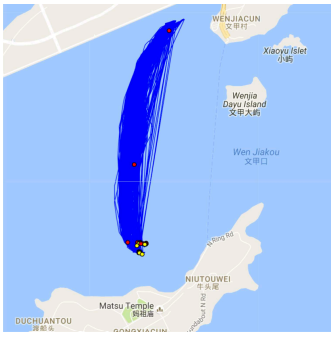
Fig. 9. ELBOW Method to Choose K. E_k with suitable K value should be the elbow point in this figure. Here, we choose $K = 33$.

The result shows that our method can extract movement patterns much better than EDR, LCSS, Hausdorff and DTW. Using our approach, the trajectories with similar moving behavior is clustered together, even if the similarity occur in different regions and time periods. Our method improved the accuracy by more than 20% than other methods. The reason why our method achieves better performance in this task is that we extract the moving behavior features and generate the trajectory moving behavior by using the whole trajectory information. However, these trajectory method use point wise (just like Hausdorff) or pair wise (EDR, LCSS, and DTW) distance to measure the distance between two trajectories.

C. Results on Vessel Motion Dataset

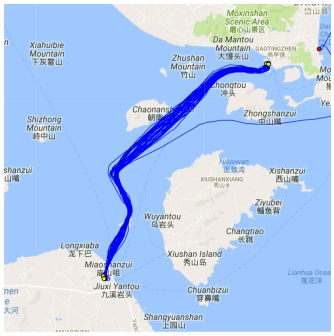
We perform two tasks on this dataset. The first one is the standard trajectory clustering task. In this task, we utilize our framework to generate clusters that have similar moving behavior, and then analyze the meaning of trajectories in them. The second one is vessel type analysis. After clustering, we examine whether the vessels having the same type are grouped into the same cluster and measure the accuracies.

Distribution of Trajectories in Cluster 1



Trajectories Nearby Matsu Temple, Fujian

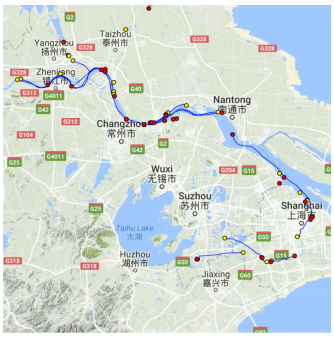
Trajectories Nearby Zhongshan, Guangdong



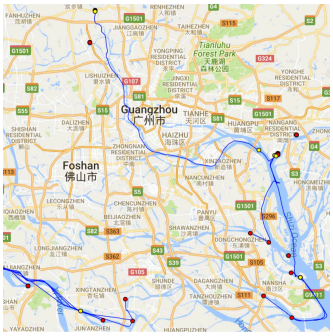
Trajectories Nearby Moxinshan, Zhoushan

Fig. 10. Trajectories in Cluster 1. The blue lines stand for the trajectories; the yellow points stand for the start point and the red points stand for the end point. Most of trajectories in this cluster are **short round trips** between **tourist cities** and generated by **passenger ships**.

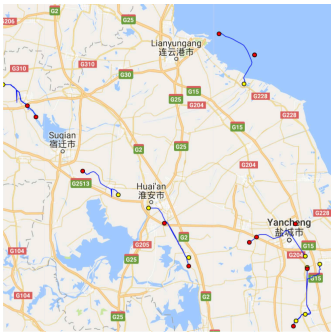
Distribution of Trajectories in Cluster 2



Trajectories in Long River Nearby Shanghai



Trajectories in Guangdong Inland River



Trajectories in Jiangsu Inland River

Fig. 11. Trajectories in Cluster 2. The blue lines stand for the trajectories; the yellow points stand for the start point and the red points stand for the end point. Most of trajectories in this cluster are distributed in **inland rivers** and it is **sparser and longer** than Cluster 1. These trajectories are generated by **inland cargo ships**.

Trajectory Clustering Task: Utilizing our framework, trajectory moving behavior vectors were generated. Most of the parameters used in this procedure are the same as synthetic dataset except the number of iterations and hidden states. Because real trajectories are usually much longer, we set iteration number to 3000 and hidden state number to 300. We use K-means to generate the clusters of Z set. K-means clustering is a method of vector quantization and widely used in cluster analysis. This algorithm aims to partition N behavior vectors of Z into k clusters in which each vector belongs to the cluster with the nearest mean. So this method is very sensitive to the value of k.

As we do not know the number of ground-truth clusters on the real data, we describe how we choose the value of K as follows. The K-means algorithm aims to find $\mathscr{C} = \{C_1, C_2, ..., C_K\}$ such that:

$$E_k = \arg\min_{C} \sum_{i=1}^{k} \sum_{Z_{TR_i}} \| Z_{TR_i} - \mu_i \|^2 \qquad (6)$$

where μ_i is the mean points in C_i. Therefore, if the value of k is good, E_k can not be very big. E_k decreases with the increase of k. We increase the number of K from 3 to 100 with step-size 5. For each K, we calculate the sum of distances from samples to their nearest centroid and denoted it as E_k. The result is shown in Figure 9. The K value corresponding to the elbow point can be found in Figure 9.

As shown in Figure 9, we choose $K = 33$. It means that there are 33 moving behavior patterns in this 4700 trajectories. Some of the cluster results are shown in Figure 10 and Figure 11. The blue lines stand for the trajectories, the yellow points stand for the start point and the red points stand for the end point. The first cluster that contains 117 trajectories is depicted in Figure 10. As shown, most of trajectories are distributed in tourist city. Besides, most of them are the short round trips. The trajectories in this cluster are generated from short passenger ship with high probability. The cluster in Figure 11 contains 180 trajectories. We can easily find that most of the trajectories are distributed in the inland river and the trajectories are sparser and longer than those in the first cluster. We examined the member trajectories in this cluster, and find most of them are generated from inland cargo ships.

The above experiments show that the clusters generated by our approach can capture the movement patterns of object in different time and space. Trajectories in these clusters are meaningful and we can easily understand the real intention of each cluster by analyzing the typical trajectory in them.

Vessel Type Analysis Task: Prior research has shown that different vessel types have different behavior patterns[22][23]. In this task, we try to recognize the vessel type by utilizing the trajectory moving behavior vectors which ate generated in trajectory clustering task.

Different vessels have different numbers of trajectories. Trajectories of vessel constitute a sequence in chronological

TABLE II
TABLE OF VESSEL TYPE CLUSTER RESULTS

	Passenger	Fishing	Cargo	Oil
Total Number	50	50	50	50
Precision	44/48=0.91	35/41=0.85	26/37=0.7	50/74=0.67
Recall	44/50=0.88	35/50=0.7	26/50=0.52	50/50=1.0
Overall Accuracy: (44+35+26+50)/200 = 0.78				

order. In order to recognize the vessel type, we took the trajectory moving behavior vectors of a vessel as the input of the sequence to sequence auto-encoder, which has been specified in Section III-B.

Similarly, we took the trajectory moving behavior vectors of an vessel as the input of encoder and minimized the mean squared error between encoder input and decoder output. Subsequently, we obtained the moving behavior vector of the vessel. Based on these vessel moving behavior vectors, we utilized clustering algorithm to get the vessel clusters. Ideally, vessels in different clusters have different vessel types. The cluster result is shown in Table 2.

Although our approach is totally unsupervised, we still observe quite good vessel typing accuracies. The over all accuracy of vessel type recognition up to 78%. Especially, the precision / recalls of the oil and passenger ship reach 0.67/1.0 and 0.91/0.88. However, the result of cargo ship is only 0.7/0.52. We consulted the experts in the shipping field for this phenomenon. The reason is that cargo ship contains many subtypes such as dry cargo ship, wet cargo ship, and roll-on-roll-off ship. These different types make a great difference in the moving behavior patterns. However, if such subtype information is available in the training process, the cluster performance is expected to have huge improvements.

In general, this set of experiments show that different vessel types have different moving behavior patterns, and our framework is good at capturing such patterns.

VI. CONCLUSION

In this paper, we proposed a novel framework for trajectory clustering with similar movement patterns. Moving behavior feature extraction algorithm was proposed to extract moving behavior features that capture space- and time- invariant characteristics of trajectories. Then, sequence to sequence auto-encoder was utilized to generate a deep representation of moving behavior sequence and address the spatio-temporal shifts problem. Moreover, we adopted many cluster algorithms to get the trajectory clusters and find which one is suited for this task.

To the best of our knowledge, it is the first attempt to use sequence to sequence auto-encoder for exploiting trajectory movement patterns. More importantly, we also propose a moving behavior feature extraction algorithm for extracting moving behavior sequence of trajectory and integrate it in our framework. We demonstrate the efficiency of our framework on both synthetic dataset and China vessel motion dataset. Experimental results show that our method has a higher accuracy than other trajectories clustering methods on simulation data and it can get useful trajectory clusters and accurately detect object groups in real-life data.

ACKNOWLEDGMENT

This work is supported by research grant NO.61472403 and No. 61303243 from National Natural Science Foundation of China (NSFC).

REFERENCES

[1] Zheng Y. Trajectory data mining: an overview[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2015, 6(3): 29.
[2] Yuan G, Sun P, Zhao J, et al. A review of moving object trajectory clustering algorithms[J]. Artificial Intelligence Review, 2016: 1-22.
[3] Besse P, Guillouet B, Loubes J M, et al. Review and perspective for distance based trajectory clustering[J]. arXiv preprint arXiv:1508.04904, 2015.
[4] Hung C C, Peng W C, Lee W C. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes[J]. The VLDB Journal, 2015, 24(2): 169-192.
[5] Lee J G, Han J, Whang K Y. Trajectory clustering: a partition-and-group framework[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 593-604.
[6] Tang W, Pi D, He Y. A Density-Based Clustering Algorithm with Sampling for Travel Behavior Analysis[C]//International Conference on Intelligent Data Engineering and Automated Learning. Springer International Publishing, 2016: 231-239.
[7] Li Z, Lee J G, Li X, et al. Incremental clustering for trajectories[C]//International Conference on Database Systems for Advanced Applications. Springer Berlin Heidelberg, 2010: 32-46.
[8] Sas C, OHare G, Reilly R. Virtual environment trajectory analysis: a basis for navigational assistance and scene adaptivity[J]. Future Generation Computer Systems, 2005, 21(7): 1157-1166.
[9] Kohonen T. The self-organizing map[J]. Neurocomputing, 1998, 21(1): 1-6.
[10] Higgs B, Abbas M. Segmentation and clustering of car-following behavior: recognition of driving patterns[J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(1): 81-90.
[11] Chao Zhang, Keyang Zhang, et al GMove: Group-Level Mobility Modeling Using Geo-Tagged Social Media. Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016
[12] Liu S, Ni L M, Krishnan R. Fraud detection from taxis' driving behaviors[J]. IEEE Transactions on Vehicular Technology, 2014, 63(1): 464-472.
[13] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(8): 1798-1828.
[14] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//Advances in neural information processing systems(NIPS). 2014: 3104-3112.
[15] Dai A M, Le Q V. Semi-supervised sequence learning[C]//Advances in Neural Information Processing Systems(NIPS). 2015: 3079-3087.
[16] Chung Y A, Wu C C, Shen C H, et al. Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Recurrent Neural Networks[J]. arXiv preprint arXiv:1603.00982, 2016.
[17] Srivastava N, Mansimov E, Salakhutdinov R. Unsupervised learning of video representations using lstms[J]. CoRR, abs/1502.04681, 2015, 2.
[18] Palangi H, Deng L, Shen Y, et al. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2016, 24(4): 694-707.
[19] Chen Q, Song X, Yamada H, et al. Learning Deep Representation from Big and Heterogeneous Data for Traffic Accident Inference[C]//Thirtieth AAAI Conference on Artificial Intelligence. 2016.
[20] Y. Bengio, P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, Neural Networks, IEEE Transactions on, vol. 5, no. 2, pp. 157166, 1994.
[21] https://en.wikipedia.org/wiki/Automatic_identification_system

[22] de Souza E N, Boerder K, Matwin S, et al. Improving Fishing Pattern Detection from Satellite AIS Using Data Mining and Machine Learning[J]. PloS one, 2016, 11(7): e0158248.
[23] Mazzarella F, Vespe M, Damalas D, et al. Discovering vessel activities at sea using AIS data: Mapping of fishing footprints[C]//Information Fusion (FUSION), 2014 17th International Conference on. IEEE, 2014: 1-7.
[24] Chen L, zsu M T, Oria V. Robust and fast similarity search for moving object trajectories[C]//Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, 2005: 491-502.
[25] Liu Y, Li Z, Xiong H, et al. Understanding of internal clustering validation measures[C]//2010 IEEE International Conference on Data Mining. IEEE, 2010: 911-916.