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 that are not space- and time- invariant. As a result, they cannot detect trajectory clusters where the within-cluster similarity occurs 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 and real data, 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 aims 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, mobility pattern mining [1], [2], location prediction and anomaly detection [3].

A plethora of trajectory clustering techniques have been proposed [4]. They typically use certain measures to quantify trajectory similarities, and then apply classic clustering algorithms (e.g., K-means, DBSCAN, spectral clustering). Popular trajectory similarity measures[5] 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 appears 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 [6] are common in many scenarios 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 extracts 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 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.
- We employ a sliding-window-based approach to extract a set of robust movement features, and then apply se-

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

quence to sequence auto-encoders to learn fixed-length representations for the trajectories. To the best of our knowledge, this is the first study that leverages recurrent neural networks 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. We overview of our method 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, trajectory pattern mining, and sequence to sequence auto-encoder.

A. Trajectory Clustering

Classic trajectory clustering approaches [4] apply distance-based or density-based clustering algorithms based on similarity measures for trajectory data [7], such as DTW (Dynamic Time Warping), EDR (Edit Distance on Real sequence) and LCSS (Longest Common Subsequences). Lee *et al.* [8] proposed a framework which first partitioned the each trajectory into sub-trajectories and then groups sub-trajectories using density-based clustering method. Tang *et al.* [9] presented a travel behavior clustering algorithm, which combined sampling with density-based clustering to deal with the noise in trajectory data. Li *et al.* [10] proposed an incremental framework to support online incremental clustering. Besse *et al.* [4] performed distance-based trajectory clustering by introducing a new distance measurement. Kohonen *et al.*[11][12] developed SOM (Self-Organizing Maps) and LVQ (Learning Vector Quantization), which could be used for adaptive trajectory analysis and clustering. While the aforementioned methods can cluster trajectories that are similar in a fixed region and period, they are inapplicable for discovering space- and time-invariant clusters.

B. Trajectory Pattern Mining

A number of methods have been proposed for mining different patterns in trajectories. Hung *et al.* [6] proposed a trajectory pattern mining framework that extracted frequent travel patterns and then trajectory routes. Zhang *et al.* [2] developed an efficient and robust method for extracting frequent sequential patterns from semantic trajectories. Higgs *et al.* [13] proposed a framework for the segmentation and clustering of car-following trajectories based on state-action variables. Zhang *et al.* [14] used the hidden Markov Model to model the mobility for different groups of users. Liu *et al.* [15] introduced a speed-based clustering method to detect taxi charging fraud behavior. Different from our work that detects general trajectory clusters, these works detect specific moving patterns in trajectory data.

C. Sequence to Sequence Auto-encoder

Sequence to sequence auto-encoder was first proposed by Sutskever *et al.* [16] for machine translation. Dai *et al.* [17] 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 fixed-length representations for videos and sentences. Specifically, Chung *et al.* [18] employed it to generate audio vector; Nitish Srivastava [19] used multilayer Long Short Term Memory (LSTM) networks to learn representations of video sequences; Hamid Palangi [20] 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 formulate our problem, then we give an overview of our framework.

A. Problem Formulation

Consider a set of moving objects $O = \{o_1, o_2, ..., o_L\}$. For each object o, its history sequence of GPS records is given by $S_o = (x_1, x_2, ..., x_M)$. Here, each x in S is a tuple (t_x, l_x, a_x, o_x) where t_x is the timestamp, l_x is a two-dimensional vector (longitude and latitude) representing the object's location, a_x is a set of attributes collected by other sensors (e.g., if the object is a car, a_x may include the speed, turning rate, fuel consumption, etc); and o_x is the object ID.

A raw sequence S_o can be sparse in practice, we segment it into a set of trajectory sequences $TR_o = (TR_1, TR_2, ..., TR_n)$, defined as follows:

Definition 1: 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 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 and also satisfy condition (1).

Figure 1 depicts a simple example of the trajectory generation process. With $S_o = (x_1, x_2, x_3, x_4, x_5, x_6, x_7))$ and $\Delta t = 3hour$, we segment the sequence into three trajectories: $TR_o = (TR_1, TR_2, TR_3)$.

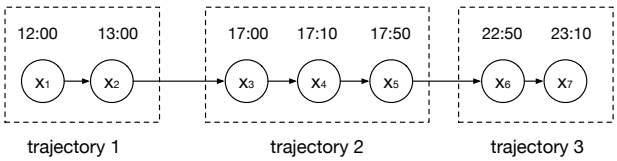

Fig. 1. Partition the sparse sequence into trajectories.

By combining the trajectories of all the objects, we obtain a trajectory set $\mathscr{T} = \{TR_1, TR_2, ..., TR_N\}$. Our goal is to generate space- and time- invariant trajectory clusters of

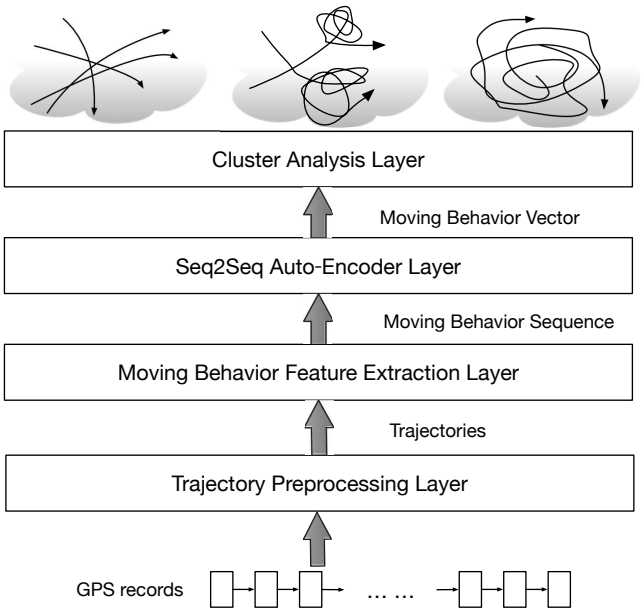

Fig. 2. Our framework for trajectory clustering.

$\mathscr{T}$. Specifically, based on the 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 appear in different geographical regions and also different parts of the trajectories.

B. Overview of The Framework

We present the framework for finding space- and time-invariant trajectory clusters in Figure 2. As shown, the framework is an unsupervised approach with four layers, detailed as followed.

- **Trajectory Preprocessing Layer:** The input of this layer is the GPS record sequences 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 the sequence into trajectories with temporal continuity.
- **Moving Behavior Feature Extraction Layer:** In this layer, all the trajectories are processed with a moving behavior feature extraction algorithm. Based on a sliding window, we transform the trajectory into a feature sequence.
- **Seq2Seq Auto-Encoder Layer:** We use a sequence to sequence auto-encoder to embed each feature sequence to a fixed-length vector. This vector encodes the 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.

IV. Methodology

In this section, we elaborate the two layers which are key in our framework: the feature extraction layer and the sequence to sequence auto-encoder layer.

A. Moving Behavior Feature Extraction

The key idea of the behavior feature extraction is to utilize a sliding window to traverse the records and extract features in each window. As shown in Figure 3, with a sliding window, we aim to obtain space- and time- invariant features to describe the moving behaviors of the object.

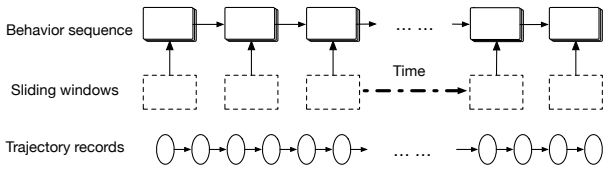

Fig. 3. Moving behavior extraction.

Let L_p and offset_p denote the width and the offset of the sliding window, respectively. While classic methods often choose $\text{offset}_p = L_p$, we find that a finer granularity of $\text{offset}_p = 1/2 \times L_p$ can effectively lead to better performance. In this way, each record in a trajectory is assigned into two windows, and most behavior changes are captured. Since the density of the records is imbalanced, some dummy windows that contain no records are also introduced, such as W_6 in Figure 4.

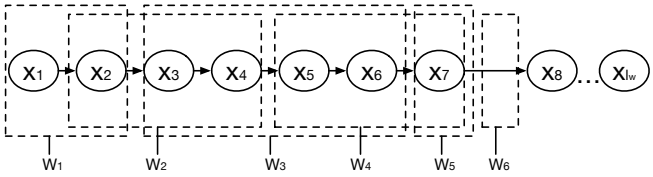

Fig. 4. Sliding time windows generation.

Now we describe the detailed feature extraction process in each sliding window as follows. The moving behavior changes can be reflected by the differences of the 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). The extracted attributes for the moving behaviors 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 ranges from 2 to R. In this way, a window with R records has $R-1$ the moving behavior attributes $(\Delta l, \Delta s, \Delta r)$.

Even if these are no attributes in the raw record, the speed and ROT also can still be calculated according to location information. As shown in Figure 5, consider a trajectory with T records $TR = (x_1, x_2 ... x_T)$. We only have the timestamp

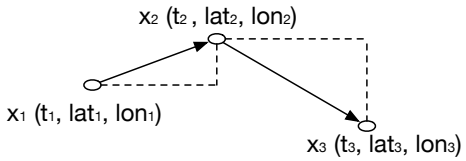

Fig. 5. Attributes completely.

and location coordinates in each record and denote them 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.

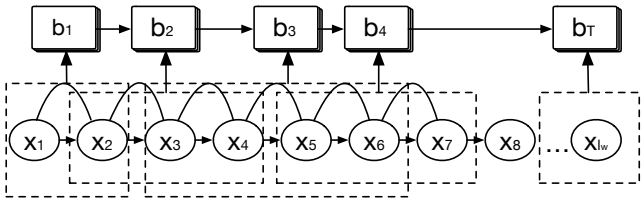

Fig. 6. The generation of moving behavior sequence.

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 $f = \{f_1, f_2, ..., f_R\}$. 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.

In summary, the moving behavior features of each 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.

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

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 $len(W.records) \geq 1$ **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}

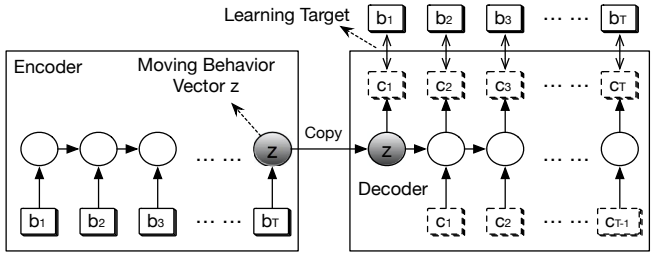

Fig. 7. Architecture of sequence to sequence auto-encoder.

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[21].

RNNs are neural networks whose hidden units form a directed cycle and it is suitable for variable length inputs. 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.

As the vanilla RNN has difficulty in learning long-term dependencies in practice[22], we use LSTM to overcome this shortage, which has been found successful in a number of applications [16][17][18][23][19][20].

The LSTM auto-encoder model is composed of two RNNs - the encoder LSTM is shown in the left part of Figure 7 and the decoder LSTM is illustrated in the right part. The input of

the model is a behavior sequence B_{TR_i}. The 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 used as the representation for the whole sequence. Then, the decoder first generates the output c_1 by taking h_T as the initialized hidden state of the decoder LSTM, and then further generate $(c_2, c_3, ..., c_T)$. The decoder LSTM is updated by:

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

The target of the decoder is to reconstruct 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 $\sum_{t=1}^{T} \parallel b_t - c_t \parallel^2$. As 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 is 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, and obtain the clusters.

V. Experiment

In this section, we empirically evaluate our method. We first introduce the datasets of the experiments and describe the compared methods. Then, we present the experiment results.

A. Dataset & Compared Methods

1) ***Dataset and Settings****:* We use both synthetic and real datasets to test the effectiveness of the framework. 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. After generating the trajectories, we computed the attributes of location with equation (1) and (2). In addition, we added Gaussian noise in the location generation process. Part of the synthetic dataset is shown in Figure 8.

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[24] is one of the most important ways for maritime domain awareness. AIS messages can be divided into dynamic messages and static messages. Dynamic messages report 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. There are totally 5,924,142 records in this dataset. After trajectory partition, we generated 4700 trajectories.

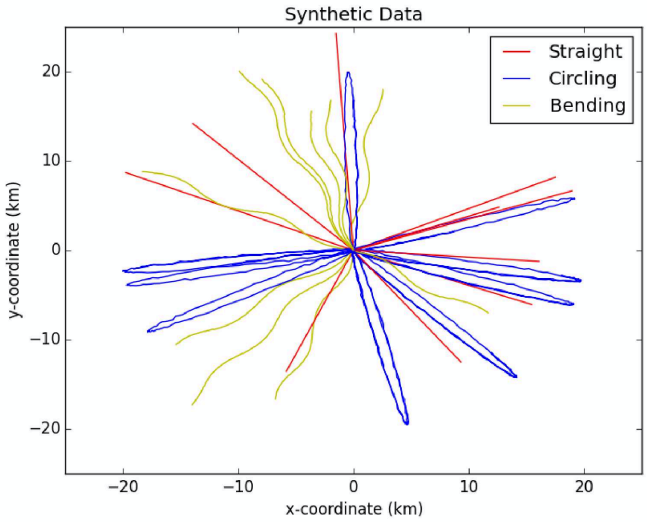

Fig. 8. Part of the synthetic data which consist of 10 straight trajectories, 10 circle trajectories and 10 bending trajectories.

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 trajectory clustering methods based on different 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 functions [5] which aim to solve the time shifting problem. They enable matching locations from different trajectories with different indexes. In contrast, Hausdorff distance is shape-based distance. For 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. For the real-life data, we tune the number of clusters and analyze the results to choose the best one.

We measure the cluster results in precision, recall and accuracy[25]. For each method, we first compute the best match between the clustering results and the groundtruth movement patterns. Then, for each movement pattern, we measure the precision and recall. The precision and recall are computed as: Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$, respectively. Here, TP (Ture Positive), stands for the number of trajectories that match the movement pattern. Finally, we measure the accuracy of each method, computed as follows: Accuracy $=$ **Sum of All TPs** / **Number of Trajectories**.

B. Results on Synthetic Dataset

For the synthetic data, we set the sliding window to 600 seconds and the offset of the window to 300 seconds. For the sequence to sequence auto-encoder procedure, we set the

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

learning rate to 0.0001 and the number of iterations to 1000. We also set the size for the LSTM cell to 100. We choose K-means algorithm to generate the trajectory clusters. EDR and LCSS need a threshold of distance to determine whether two records are matching. After tuning, we set the threshold to 100 meters. The clustering performance of different methods is shown in Table 1.

TABLE I
CLUSTERING PERFORMANCE ON SYNTHETIC DATA.

	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.

The results show that our method can extract movement patterns much better than EDR, LCSS, Hausdorff and DTW. Using our approach, the trajectories with similar moving behaviors are clustered together, even if the similarity occur in different regions and time periods. Our method has improved the accuracy by more than 20% than other methods.

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 or not and measure the accuracies.

Trajectory Clustering Task: Utilizing our framework, trajectory moving behavior vectors are 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 the number of iterations to 3000 and the size of the hidden layer to 300. We use K-means to generate the clusters for the Z set.

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. 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 $K = 33$, extracting 33 clusters for the 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. We find that the trajectories

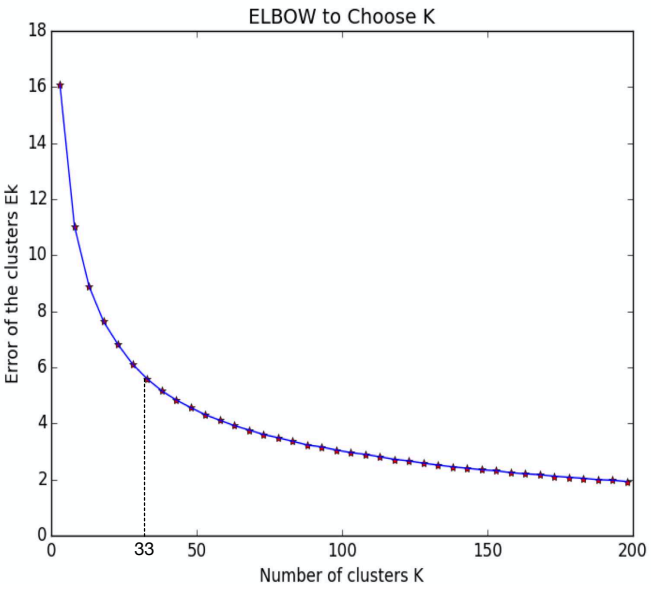

Fig. 9. ELBOW Method to Choose K. E_k with suitable K value should be the elbow point in this figure. Here, we choose $K = 33$.

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**.

in this cluster are mostly 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

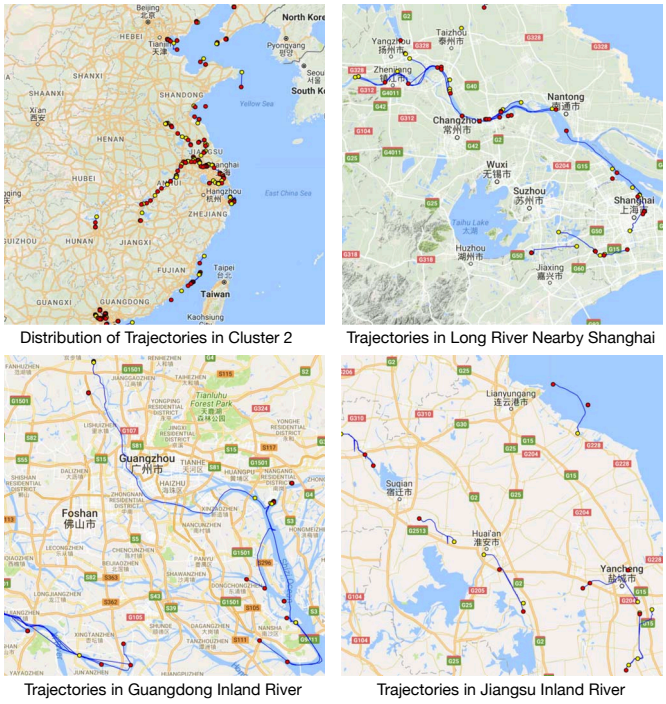

Fig. 11. Trajectories in Cluster 2. The blue lines stand for the trajectories; the yellow points stand for the start point and the red points stand for the end point. Most trajectories in this cluster are distributed in **inland rivers** and it is **sparser and longer** than Cluster 1. These trajectories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

member trajectories in this cluster, and find that 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 the objects in different time and space. The trajectories in each cluster are meaningful and we can easily interpret each cluster by analyzing the typical trajectories in them.

Vessel Type Analysis Task: Prior research has shown that different vessel types have different behavior patterns[26][27]. In this task, we try to recognize the vessel type by utilizing the trajectory clustering.

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 our clustering algorithm to get the vessel clusters. Ideally, the vessels in different clusters should have different vessel types. The clustering accuracy results are shown in Table 2.

Although our approach is totally unsupervised, we still observe quite good vessel typing accuracies. The overall accuracy for vessel type recognition is about 78%. Especially, the precision / and recall for the oil ship and the passenger ship are 0.67/1.0 and 0.91/0.88, respectively. 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 ships contain many subtypes such as dry cargo ship,

TABLE II
VESSEL TYPE CLUSTERING 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				

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 better 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. A moving behavior feature extraction algorithm was proposed to extract moving behavior features that captured space- and time- invariant characteristics of trajectories. Then, the sequence to sequence auto-encoder was utilized to generate a deep representation of moving behavior sequence and address the spatio-temporal shifts problem. We have demonstrated the effectiveness of our framework on both synthetic and real datasets. Experimental results show that our method has a higher accuracy than other trajectories clustering methods on synthetic data. Additionally, it can get useful trajectory clusters and accurately detect object groups for the real data.

ACKNOWLEDGMENT

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

REFERENCES

[1] Q. Yuan, W. Zhang, C. Zhang, X. Geng, G. Cong, and J. Han, "PRED: periodic region detection for mobility modeling of social media users," in *WSDM*. ACM, 2017, pp. 263–272.
[2] C. Zhang, J. Han, L. Shou, J. Lu, and T. F. L. Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *PVLDB*, vol. 7, no. 9, pp. 769–780, 2014.
[3] Y. Zheng, "Trajectory data mining: An overview," *ACM TIST*, vol. 6, no. 3, pp. 29:1–29:41, 2015.
[4] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artif. Intell. Rev.*, vol. 47, no. 1, pp. 123–144, 2017.
[5] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Review and perspective for distance based trajectory clustering," *CoRR*, vol. abs/1508.04904, 2015.
[6] C. Hung, W. Peng, and W. Lee, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *VLDB J.*, vol. 24, no. 2, pp. 169–192, 2015.
[7] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *SIGMOD Conference*. ACM, 2005, pp. 491–502.
[8] J. Lee, J. Han, and K. Whang, "Trajectory clustering: a partition-and-group framework," in *SIGMOD Conference*. ACM, 2007, pp. 593–604.

[9] W. Tang, D. Pi, and Y. He, "A density-based clustering algorithm with sampling for travel behavior analysis," in *IDEAL*, ser. Lecture Notes in Computer Science, vol. 9937. Springer, 2016, pp. 231–239.
[10] Z. Li, J. Lee, X. Li, and J. Han, "Incremental clustering for trajectories," in *DASFAA (2)*, ser. Lecture Notes in Computer Science, vol. 5982. Springer, 2010, pp. 32–46.
[11] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.
[12] C. Sas, G. M. P. O'Hare, and R. Reilly, "Virtual environment trajectory analysis: a basis for navigational assistance and scene adaptivity," *Future Generation Comp. Syst.*, vol. 21, no. 7, pp. 1157–1166, 2005.
[13] B. Higgs and M. M. Abbas, "Segmentation and clustering of car-following behavior: Recognition of driving patterns," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 1, pp. 81–90, 2015.
[14] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "Gmove: Group-level mobility modeling using geo-tagged social media," in *KDD*. ACM, 2016, pp. 1305–1314.
[15] S. Liu, L. M. Ni, and R. Krishnan, "Fraud detection from taxis' driving behaviors," *IEEE Trans. Vehicular Technology*, vol. 63, no. 1, pp. 464–472, 2014.
[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.
[17] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *NIPS*, 2015, pp. 3079–3087.
[18] Y.-A. Chung, C.-C. Wu, C.-H. Shen, and H.-Y. Lee, "Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks," in *Proc. Interspeech*, 2016.
[19] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 843–852.
[20] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. K. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 24, no. 4, pp. 694–707, 2016.
[21] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.
[22] Y. Bengio, P. Y. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
[23] Q. Chen, X. Song, H. Yamada, and R. Shibasaki, "Learning deep representation from big and heterogeneous data for traffic accident inference," in *AAAI*. AAAI Press, 2016, pp. 338–344.
[24] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang, "Automatic identification system (ais): data reliability and human error implications," *Journal of navigation*, vol. 60, no. 03, pp. 373–389, 2007.
[25] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *ICDM*. IEEE Computer Society, 2010, pp. 911–916.
[26] E. N. de Souza, K. Boerder, S. Matwin, and B. Worm, "Improving fishing pattern detection from satellite ais using data mining and machine learning," *PloS one*, vol. 11, no. 7, p. e0158248, 2016.
[27] F. Mazzarella, M. Vespe, D. Damalas, and G. Osio, "Discovering vessel activities at sea using AIS data: Mapping of fishing footprints," in *FUSION*. IEEE, 2014, pp. 1–7.