Received: 15 March 2017 | Revised: 5 July 2017 | Accepted: 8 September 2017
DOI: 10.1111/exsy.12252

WILEY Expert Systems

SPECIAL ISSUE PAPER

Learning deep representation for trajectory clustering

Di Yao[1,5] | Chao Zhang[2] | Zhihua Zhu[1,5] | Qin Hu[3] | Zheng Wang[4] | Jianhui Huang[1,5] | Jingping Bi[1,5]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[3]College of Information Science and Technology, Beijing Normal University, Beijing, China
[4]School of Information Technologies, University of Sydney, Sydney, Australia
[5]University of Chinese Academy of Sciences, Beijing, China

Correspondence
Jingping Bi, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
Email: bjp@ict.ac.cn

Funding information
National Natural Science Foundation of China (NSFC), Grant/Award Number: 61472403 and 61303243;

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 such as location prediction and activity recognition. Although a plethora of trajectory clustering techniques have been proposed, they often rely on spatio-temporal 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 behaviour 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.

KEYWORDS
recurrent neural network, representation learning, sequence-to-sequence learning, trajectory clustering

1 | INTRODUCTION

Owing to the rapid growth of Global Positioning System (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 (Q. Yuan et al., 2017; Zhang, Han, Shou, & Lu, 2014), location prediction, and anomaly detection (Zheng, 2015).

A plethora of trajectory clustering techniques have been proposed (G. Yuan, Sun, Zhao, Li, & Wang, 2017). They typically use certain measures to quantify trajectory similarities and then apply some classic clustering algorithms (e.g., K-means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and spectral clustering) to detect clusters. And popular trajectory similarity measures (Besse, Guillouet, Loubes, & Royer, 2016) include dynamic time warping (DTW), edit distance on real sequence (EDR), and longest common subsequences (LCSS). Although these measures can group trajectories that are similar to each other in a particular geographical region and time period, many practical applications work with trajectories that distribute in different regions with different time spans and sampling rates. In such applications, one is often required to identify clusters, each of which represents similarity of trajectories regardless of the differences in time and space. For example, the taxis in traffic jams can have similar moving behaviours, but the traffic jams usually occur in different areas in the city with different durations. Such spatio-temporal shifts (Hung, Peng, & Lee, 2015) are common in many scenarios and render current trajectory clustering algorithms ineffective.

In this work, we revisit the trajectory clustering problem and develop a method to 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 natural language processing. Given the input trajectories, our goal is to convert each trajectory into a fixed-length representation that is able to well encode the object's moving behaviours. Once the high-quality trajectory representations are learnt, any classic clustering algorithms can be easily applied based on practical needs.

Nevertheless, it is non-trivial to directly apply RNNs to the input trajectories to obtain representations of high qualities because of the varying qualities and sampling frequencies of the given trajectories. Furthermore, we find that a naive strategy that considers each trajectory as a sequence of three-dimensional records (time, latitude, and longitude) leads to dramatically oscillating parameters and nonconvergence 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 its space- and time-invariant features of the trajectories. With extracted features, 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 contribute 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 involves grouping 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 sequence-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 RNNs for the moving behaviour analysis and trajectory clustering task on GPS data.
- 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 2, we investigate some related work. We offer the problem formulation and the method overview in Section 3. And then, we detail the main steps of the methods in Section 4. We empirically evaluate the proposed method in Section 5 and finally conclude in Section 6.

2 | RELATED WORKS

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

2.1 | Trajectory clustering

Classic trajectory clustering approaches (G. Yuan et al., 2017) always utilize distance-based or density-based clustering algorithms based on similarity measures for trajectory data (L. Chen, Özsu, & Oria, 2005), such as DTW, EDR, and LCSS. Lee, Han, and Whang (2007) proposed a framework that first partitioned each trajectory into several subtrajectories and then grouped them using density-based clustering method. Tang, Pi, and He (2016) presented a travel behaviour clustering algorithm, combining sampling with density-based clustering to deal with the noise in trajectory data. In Li, Lee, Li, and Han (2010), the author put forward an incremental framework to support online incremental clustering. And Besse et al. (2016) performed distance-based trajectory clustering by introducing a new distance measurement. Kohonen (1998) and Sas, O'Hare, and Reilly (2005) developed self-organizing maps (SOM) and learning vector quantization for adaptive trajectory analysis and clustering. Moreover, trajectory clustering can be used for gesture recognition (Alahi, Vignesh Ramanathan, Robicquet, Li, & Savarese, 2016; Keskin, Cemgil, & Akarun, 2011). Although the aforementioned methods can only cluster trajectories that are similar in a fixed region and period, they are inapplicable for discovering space- and time-invariant clusters.

2.2 | Trajectory pattern mining

A number of methods have been proposed for mining different patterns in trajectories. Hung et al. (2015) proposed a trajectory pattern mining framework that extracted frequent travel patterns and trajectory routes. Zhang et al. (2014) developed an efficient and robust method for extracting frequent sequential patterns from semantic trajectories. Higgs and Abbas (2015) presented a framework for the segmentation and clustering of car-following trajectories based on state-action variables. Zhang et al. (2016) used the Hidden Markov Model to formulate the mobility for different groups of users. S. Liu, Ni, and Krishnan (2014) introduced a speed-based clustering method to detect taxi-charging fraud behaviour. Different from the above research on specific moving patterns in trajectory data, we plan to detect general clusters.

2.3 | Sequence-to-sequence auto-encoder

Sequence-to-sequence auto-encoder was first proposed by Sutskever, Vinyals, and Le (2014) for machine translation. Dai and Le (2015) 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 effectiveness of sequence-to-sequence auto-encoders for generating fixed-length representations for videos and sentences. Specifically, Chung, Wu, Shen, and Lee (2016) employed it to generate audio vector; Srivastava, Mansimov, and Salakhutdinov (2015) used multilayer long short term memory (LSTM) networks to learn representations of video sequences; Palangi et al. (2016) generated a deep sentence embedding for information retrieval. However, we are not aware of any previous works that apply auto-encoders to GPS 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.

3 | PROBLEM FORMULATION AND GENERAL FRAMEWORK

In this section, we first formulate our problem, and then we give an overview of our framework.

3.1 | Problem formulation

Given a set of moving objects $O = \{o_1, o_2, \dots, o_L\}$, each object o_i has a history sequence of GPS records $S_o = (x_1, x_2, \dots, x_M)$. Here, each element x is consisted of 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, and fuel consumption); and o_x is the object ID.

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

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

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 $\Delta t = 3$ hr, we segment the sequence into three trajectories: $TR_o = (TR_1, TR_2, TR_3)$.

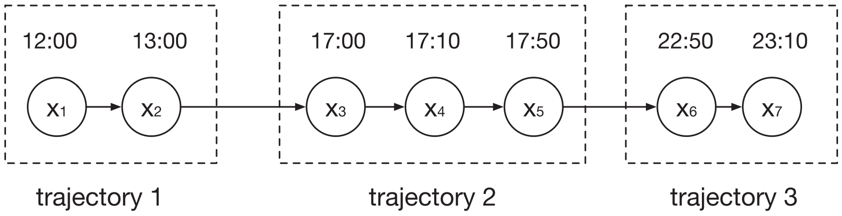

FIGURE 1 Partition the sparse sequence into trajectories

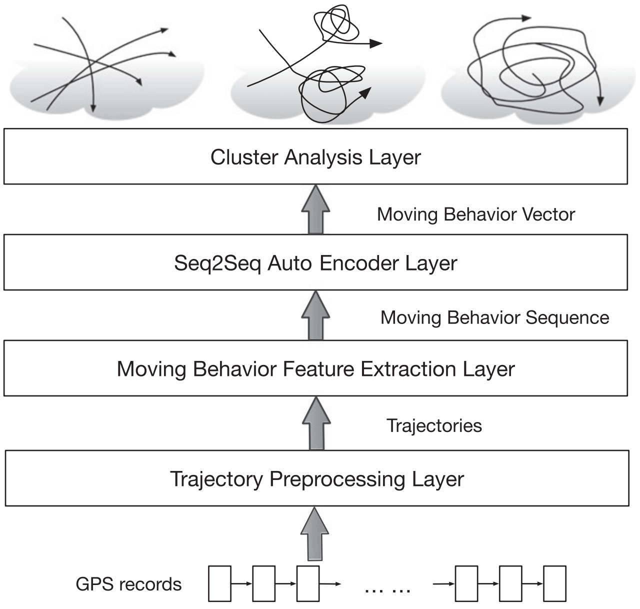

FIGURE 2 Our framework for trajectory clustering. GPS = Global Positioning System

Combining the trajectories of all the objects, we can obtain a trajectory set $\mathcal{T} = \{TR_1, TR_2, \dots, TR_N\}$. Our goal is to generate space- and time-invariant trajectory clusters of $\mathcal{T}$. Specifically, based on the objects' movement patterns, we need to generate a set of clusters $\mathcal{O} = \{C_1, C_2, \dots, C_K\}$. In each cluster, the similarity shared by the member trajectories may appear in different geographical regions and also different parts of the trajectories.

3.2 | Overview of the framework

We present the framework for finding space- and time-invariant trajectory clusters in Figure 2. Generally, the framework is an unsupervised approach with four layers:

- Trajectory preprocessing layer: The input of this layer is the GPS record sequences of the moving objects. Note that the sequence is noisy and the temporal intervals between some record pairs can be very large. In this layer, we remove the low-quality GPS records and cut the sequences into trajectories with temporal continuity (detailed in Section 3.1).
- Moving behaviour feature extraction layer: In this layer, all the trajectories are processed with a moving behaviour feature extraction algorithm. Based on a sliding window, we transform each 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 that encodes the movement pattern of the trajectory.
- Cluster analysis layer: Finally, we choose a classic clustering algorithm based on the practical needs and group the learnt representations into clusters.

4 | METHODOLOGY

In this section, we elaborate two main layers in our framework: the feature extraction layer and the sequence-to-sequence auto-encoder layer. Additionally, we offer an overview of typical recurrent units that is used in our method.

4.1 | Moving behaviour feature extraction

The basic idea of the behaviour feature extraction is to utilize a sliding window to traverse the records and obtain features in each window. As shown in Figure 3, with the help of sliding window, we aim to acquire space- and time-invariant features to describe the moving behaviours of the object.

Let L_p and offset$_p$ denote the width and the offset of the sliding window, respectively. Note that classic methods often set offset$_p = L_p$, but we find that a finer granularity of offset$_p = 1/2 \times L_p$ can lead to better performance. In this way, each record in a trajectory is assigned into two consecutive windows, and most behaviour changes can be captured. Because the record density is non-uniform, some dummy windows with no records are also introduced, such as W_6 in Figure 4.

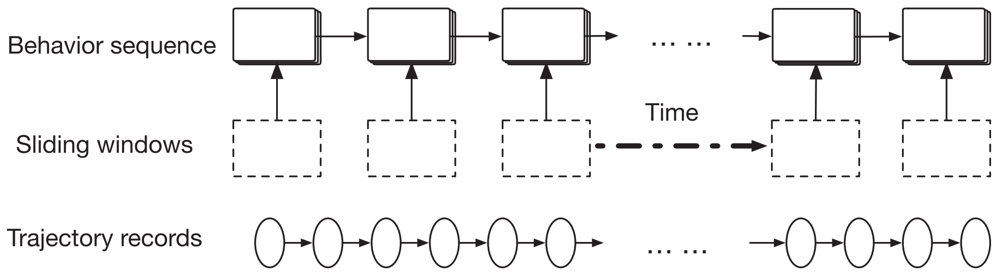

FIGURE 3 Moving behaviour extraction

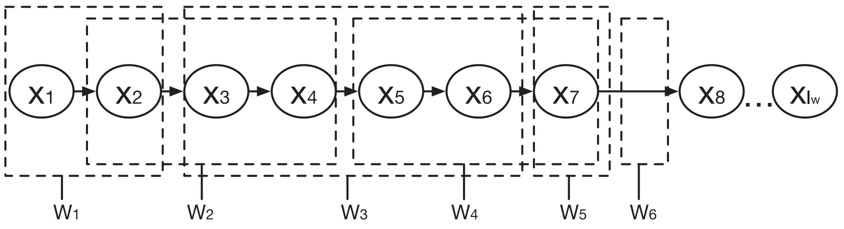

FIGURE 4 Sliding time windows generation

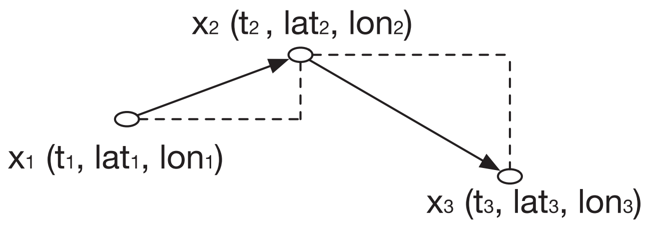

FIGURE 5 Attributes completely

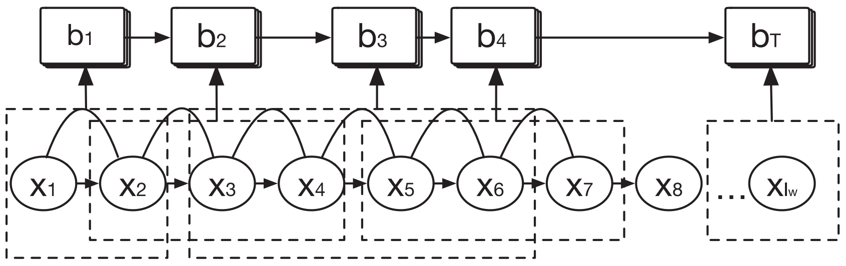

FIGURE 6 The generation of moving behaviour sequence

Now, we describe the detailed feature extraction process in each sliding window as follows. The moving behaviour changes can be reflected by the differences of the attributes between two consecutive records. Suppose a window contains R records, denoted as $W = (x_1, x_2, \dots, x_R)$, and the attributes in each record consist of speed and rate of turn (ROT). Then the extracted attributes of the moving behaviours 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 $R - 1$ moving behaviour attributes $(\Delta l, \Delta s, \Delta r)$.

Even though the speed and ROT are not included as explicit attributes in the raw record, they can be calculated according to the location information. As shown in Figure 5, consider a trajectory with T records $TR = (x_1, x_2 \dots x_T)$. Each of them only has the timestamp and location coordinates, denoted as (t, lat, lon). For the first record 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

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

and

$$r_{x_i} = \arctan \frac{lon_{x_i} - lon_{x_{i-1}}}{lat_{x_i} - lat_{x_{i-1}}}, \tag{2}$$

where $i = 2, \cdots, T$. After this procedure, the speed and ROT can be derived for each trajectory.

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 $f_i = (f_{\Delta l_i}, f_{\Delta s_i}, f_{\Delta r_i})$, where $f_{\Delta l_i} = \Delta l_i / \Delta t_i$, $f_{\Delta s_i} = \Delta s_i$ and $f_{\Delta r_i} = \Delta r_i$. For two consecutive records, $f_{\Delta l_i}$ denotes the average speed, $f_{\Delta s_i}$ stands for the change of speeds, and $f_{\Delta r_i}$ represents the change of ROTs. After computing these features in each pair, we get a feature set $f = \{f_1, f_2, \dots, f_R\}$. Next, we use the statistics 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 behaviour features of each window b has $3 \times 6 = 18$ dimensions, consisted 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. Figure 6 and Algorithm 1 show the generation procedure of moving behaviour feature sequence.

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

Algorithm 1 Behaviour feature extraction algorithm

```
Input:
    GPS records for a trajectory $TR$
Output:
    The behaviour sequence of trajectory $TR$, $B_{TR}$
1:  Initialize $B_{TR} = []$
2:  $windows = sliding\_windows(TR)$
3:  for each window $W$ in $windows$ do
4:    if $len(W.records) \geq 1$ then
5:      Initialize $F_W = []$
6:      for each record $r_i$ in $W.records$ do
7:        $r_{i-1} = find\_pre(r_i)$
8:        $F_i = compute\_features(r_i, r_{i-1})$
9:        $F_W.add(F_i)$
10:     end for
11:     $B_W = generate\_behavior(F_W)$
12:     $B_{TR}.add(B_W)$
13:   end if
14: end for
15: return $B_{TR}$
```

4.2 | LSTM and GRU recurrent neural networks

RNNs are apt at modelling input sequences with variable lengths, the recurrent units of which form a directed cycle. For an input sequence $X = (x_1, x_2, \dots, x_T)$, where $i \in [1, N]$, RNNs update its hidden state h_t according to the current input x_t and the previous hidden state h_{t-1}. The hidden state h_t acts as an internal memory at time t that captures dynamic temporal information in the sequence. In specific, the hidden state is updated as

$$h_t = f(h_{t-1}, x_t), \tag{3}$$

where $f(\cdot)$ is the activation function. Note that function $f(\cdot)$ is used to compute a weighted sum of inputs and apply a nonlinear transformation, which depends on the type of recurrent units .

As the vanilla RNN has difficulty in learning long-term dependencies in practice (Bengio, Simard, & Frasconi, 1994), we employ both LSTM and gated recurrent unit (GRU; Cho, van Merrienboer, Bahdanau, & Bengio, 2014) to overcome this shortage. We briefly describe the updating schemes in LSTM and GRU below.

4.2.1 | LSTM unit

Given an input sequence $X = (x_1, x_2, \dots, x_T)$, LSTM computes the hidden vector sequence $H = (h_1, h_2, \dots, h_T)$ by maintaining a memory of h_{t-1} at time t and deciding whether to keep the existing memory,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{4}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{6}$$

$$\tilde{c}_t = \tanh(W_{\tilde{c}} \cdot [h_{t-1}, x_t] + b_{\tilde{c}}), \tag{7}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \text{ and} \tag{8}$$

$$h_t = o_t \cdot \tanh(c_t), \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function; c_t is the memory content of the unit; and $\tilde{c}_t$ stands for the new memory content. There are three gates in LSTM: the input gate i_t, the forget gate f_t, and the output gate o_t. Note that i_t modulates the extent to which the new memory should be stored; the forget gate f_t modulates the degree to which the existing memory should be forgotten; and o_t modulates the amount of memory content for exposure. In the above, the memory content c_t at time t is updated by Equation 8. After that, the hidden state h_t is updated according to Equation 9. Intuitively, LSTM detects the important part of input and stores it for a long time. Hence, it is suitable for learning long-term dependence that has been proved successful in a number of applications (Q. Chen, Song, Yamada, & Shibasaki, 2016; Chung et al., 2016; Dai & Le, 2016; Srivastava et al., 2015; Sutskever et al., 2014).

4.2.2 | GRU unit

GRU is widely adopted in encoder- and decoder-based machine translation (Cho et al., 2014). It allows each recurrent unit to adaptively capture the dependencies of different time scales. Different from LSTM, GRU controls the flow of information inside the unit without a separate memory cell. More formally

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{10}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \tag{11}$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t \cdot h_{t-1}, x_t]), \text{ and} \tag{12}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t. \tag{13}$$

GRU has only two gates: the update gate z_t and the reset gate r_t. The hidden state h_t is a linear interpolation of h_{t-1} and $\tilde{h}_t$. As the combination of input gate and forget gate in LSTM, the update gate z_t modulates how much the unit stores or forgets new information. In this way, GRU merges the memory content c into the hidden state and achieves a simpler architect design.

4.3 | Seq2Seq auto-encoder

In this section, we describe a model utilizing sequence-to-sequence auto-encoder to reconstruct the moving behaviour sequence and generating a fixed-length deep representation of the trajectory (Bengio, Courville, & Vincent, 2013).

The sequence-to-sequence auto-encoder model is composed of two RNNs—The encoder RNN shown in the left part of Figure 7, and the decoder RNN is illustrated in the right side. The input of the model is a behaviour sequence $B_{TR_i} = (b_1, b_2, \ldots, b_T)$. The encoder RNN reads the input sequence sequentially and update hidden state h_t accordingly. The encoder RNN is updated by

$$h_t = f(h_{t-1}, b_t). \tag{14}$$

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 RNN and further generate $(c_2, c_3, \ldots, c_T)$. The decoder RNN is updated by

$$h^d{}_t = f(h^d{}_{t-1}, c_{t-1}, h_T). \tag{15}$$

The target of the decoder is to reconstruct the input sequence $B_{TR_i} = (b_1, b_2, \ldots, b_T)$. In other words, the encoder RNN and decoder RNN are trained together by minimizing the reconstruction error, measured by the general mean squared error:

$$MSE = \sum_{t=1}^{T} \|b_t - c_t\|^2. \tag{16}$$

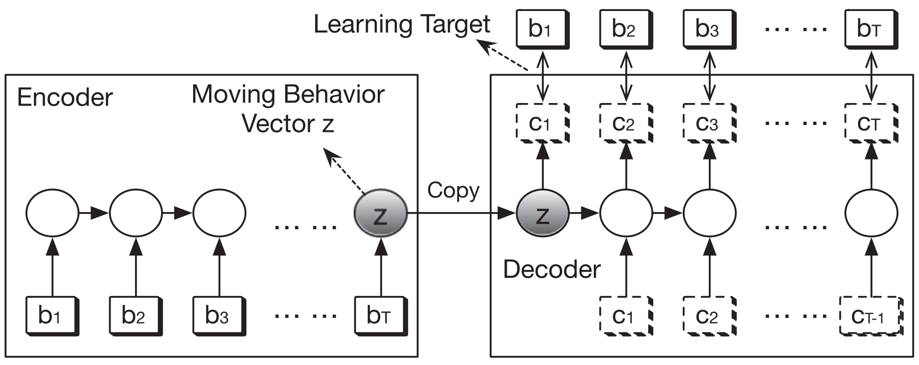

FIGURE 7 Architecture of sequence-to-sequence auto-encoder

As the input sequence is taken as the learning target, the training process does not need any labelled data. The fixed-length moving behaviour vector z is a meaningful representation for the input behaviour sequence B_{TR_i}, because the hole input sequence can be reconstructed from z by the decoder.

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

5 | EXPERIMENT

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

5.1 | Data set and compared methods

5.1.1 | Data set and settings

We use both synthetic and real data sets to test the effectiveness of the framework. For the synthetic data set, we simulated 9,000 trajectories including nine kinds of movement patterns that consist of three kinds of basic movement patterns {*Straight*, *Circling*, *Bending*} and six kinds of combination patterns {*Straight* + *Circling*, *Straight* + *Bending*, *Circling* + *Bending*, *Circling* + *Straight*, *Bending* + *Straight*, *Bending* + *Circling*}. Each pattern

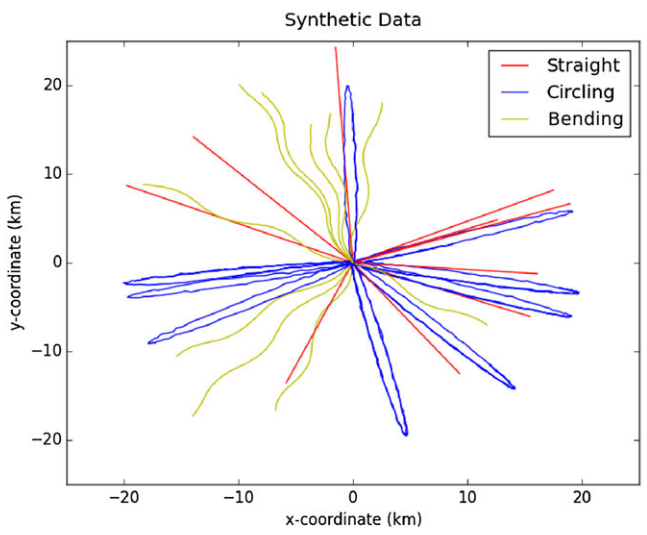

FIGURE 8 Part of the synthetic basic movement patterns that consist of 10 straight trajectories, 10 circle trajectories, and 10 bending trajectories

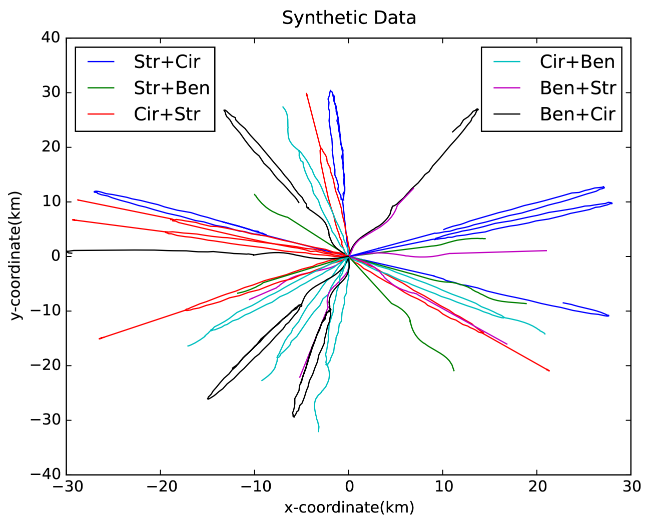

FIGURE 9 Part of the synthetic combine movement patterns. Each combine pattern has five trajectories. Str = Straight; Cir = Circling; Ben = Bending

has 1,000 trajectories. The sampling frequency and time length of each trajectory were generated randomly from 2,500 to 5,000 s. After generating the trajectories, we computed the attributes of location with Equations 1 and 2. In addition, we added Gaussian noise in the location generation process. Part of the synthetic data set is shown in Figures 8 and 9.

The real data set corresponds to 200 vessels in China, containing 50 cargo ships, 50 fishing ships, 50 oil ships, and 50 passenger ships. The vessel motion data are collected by Automatic Identification System (AIS). AIS (Harati-Mokhtari, Wall, Brooks, & Wang, 2007) that 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, including the time, position (longitude and latitude), course over ground, speed over ground, and heading. Static messages include type, name, and size. The recording time is from May 2016 to June 2016. There are totally 5,924,142 records in the data set. After trajectory partition, we generated 4,700 trajectories.

5.1.2 | Compared methods

For the parameter setting, both LSTM and GRU have three major parameters: (a) the learning rate α that controls the parameter updating step size; (b) the size of the hidden state m, which controls the trajectory embedding size; and (c) the number of training epochs n. Allowing for the influence of

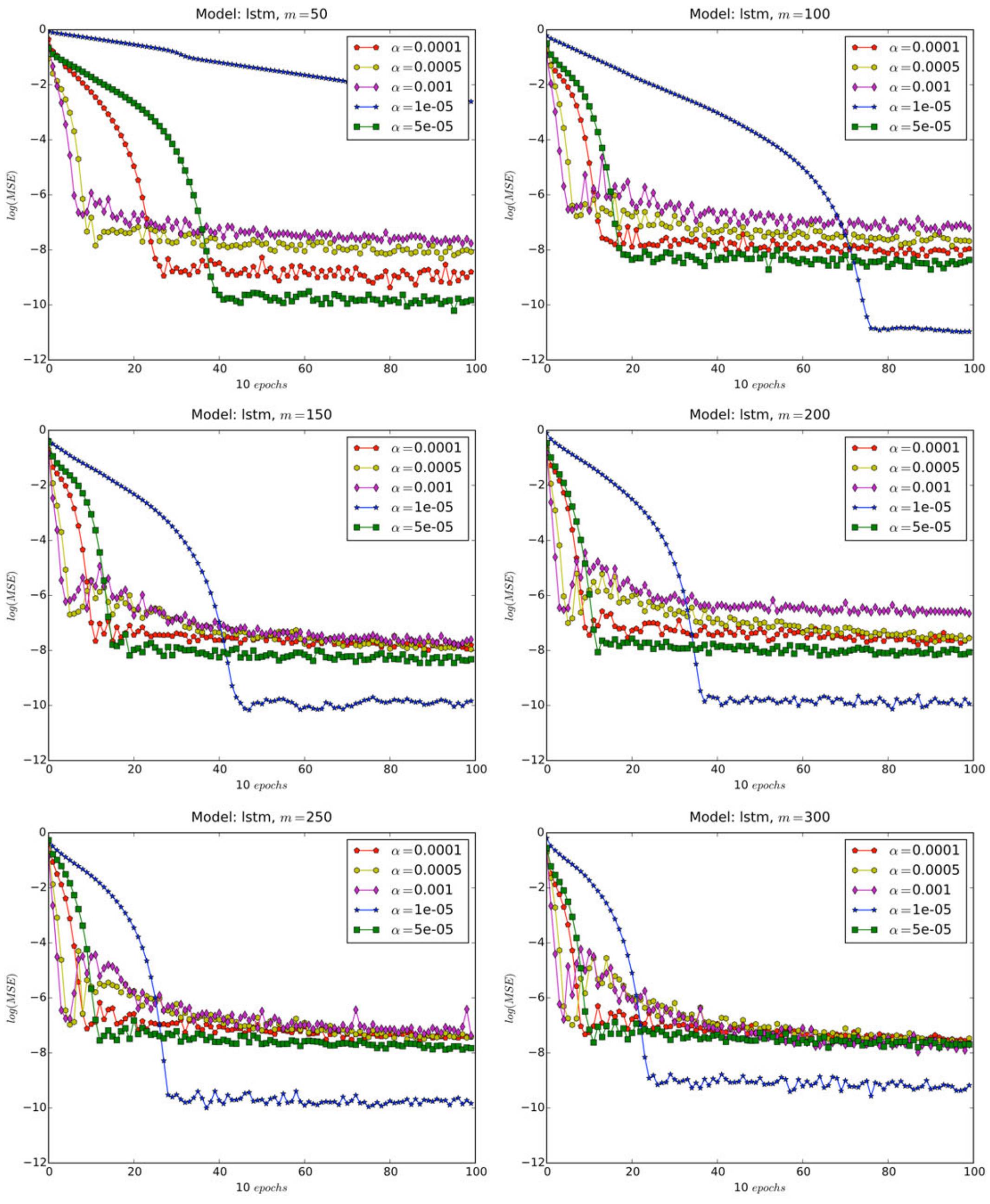

FIGURE 10 Long short term memory (LSTM) $log(MSE)$ changes with different parameters

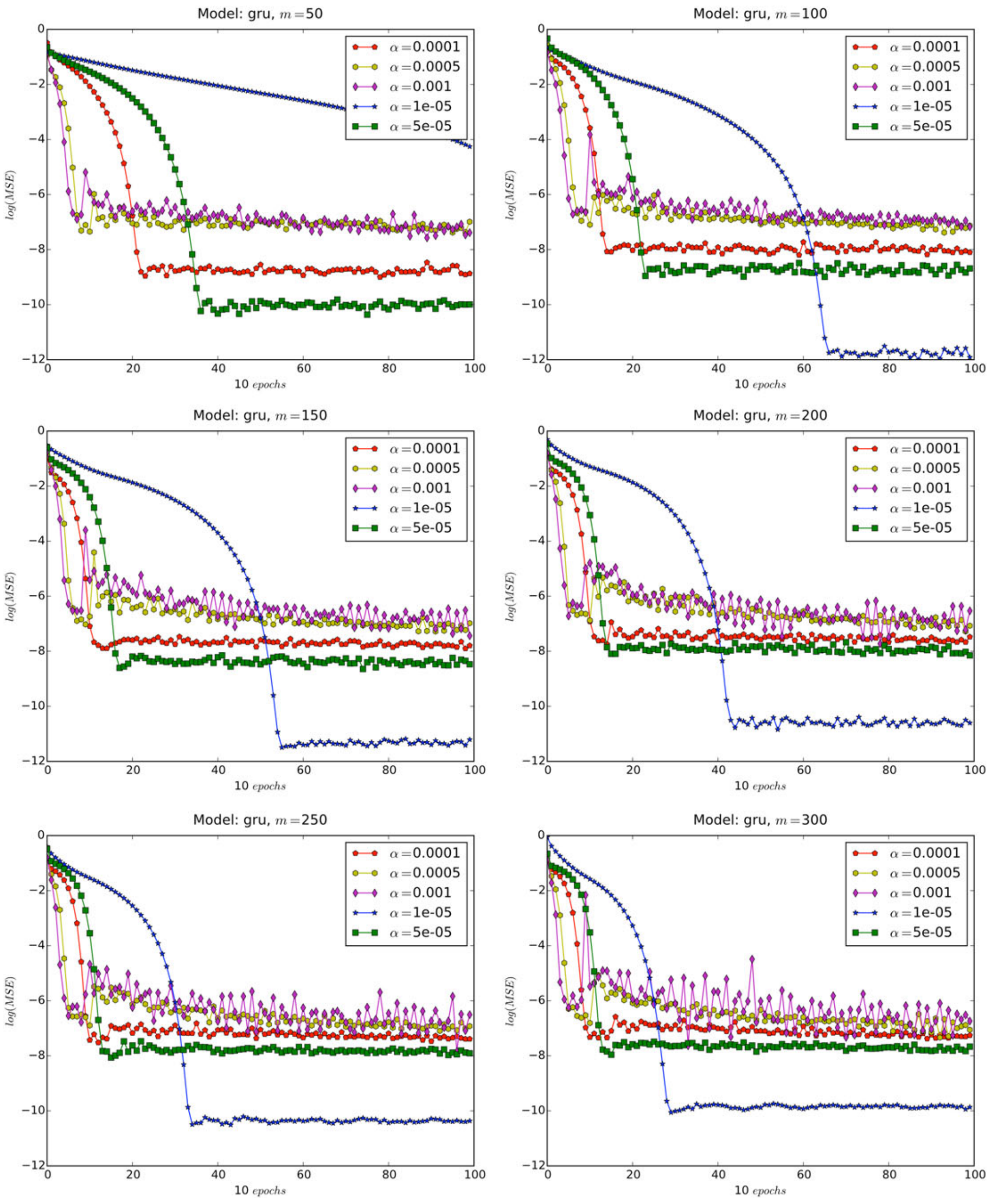

FIGURE 11 Gated recurrent unit (GRU) $log(MSE)$ changes with different parameters

different parameters, we set the default values of the parameters as follows: (a) $\alpha = 0.00001$, $m = 250$, and $n = 400$ for LSTM; and (b) $\alpha = 0.00001$, $m = 100$, and $n = 800$ for GRU. For the behaviour extraction layer, we set the sliding window as 600 s and the offset of the window as 300 s.

We implemented the framework with Python and TensorFlow. All the experiments were performed on a server with Intel Xeon CPU 2.10 GHz. The data and code are publicly available.[1]

We compare our method, including LSTM and GRU, 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 (Besse et al., 2016) that aim to solve the time-shifting problem. They are able to match locations from different trajectories with different indexes. In contrast, Hausdorff distance is shape-based distance. For each measure, we choose K-Medoids as the clustering algorithm. For the synthetic data, because the number of moving behaviour patterns are known, we set the number of clusters as 9. Whereas for the real-life data, we tune the number of clusters and analyse the corresponding results to choose the best.

We measure the cluster results in precision, recall, and accuracy (Y. Liu, Li, Xiong, Gao, & Wu, 2010). For each method, we first compute the best match between the clustering results and the ground-truth movement patterns. Then, for each movement pattern, we measure the precision and recall. The precision p and recall r are computed as

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

$$p = \frac{TP}{TP + FP}, r = \frac{TP}{TP + FN}. \tag{17}$$

Here, true positive (TP) 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**.

5.2 | Parameters analysis

We now study the effects of different parameters: the learning rate α, the hidden state size m, and the number of training epochs n. We tune the parameters as follows: $\alpha = [0.00001, 0.00005, 0.0001, 0.0005]$, $m = [50, 100, 150, 200, 250, 300]$, and $n = 1,000$. When studying the effect of one parameter, we fix the other parameters to their default values. For different settings, we measure the training error

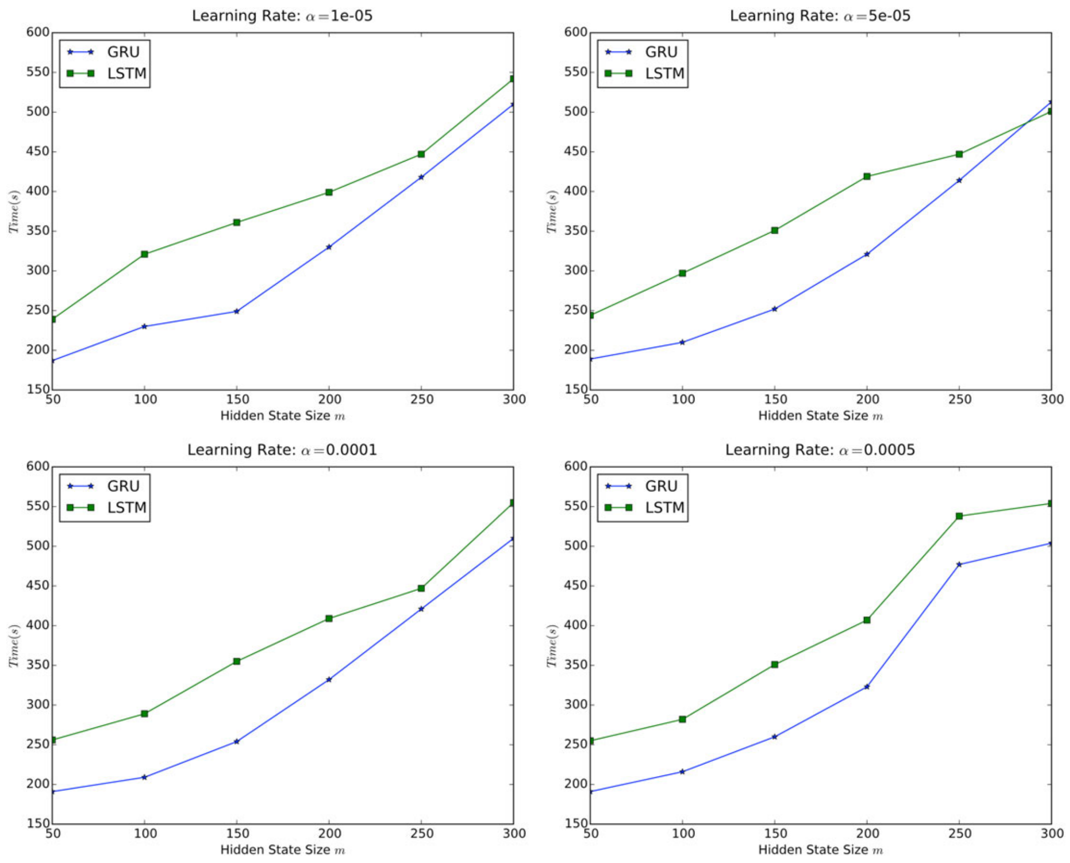

FIGURE 12 Time-consuming changes with different parameters. LTSM = long short term memory; GRU = gated recurrent unit

TABLE 1 Clustering performance on synthetic data

	EDR	LCSS	DTW	Hausdorff	GRU-s2s	LSTM-s2s
Straight	0.465/0.563	0.460/0.411	0.411/0.613	0.423/0.263	0.643/0.723	0.760/0.703
Circling	0.550/0.482	0.610/0.643	0.540/0.462	0.415/0.531	0.768/0.756	0.766/0.823
Bending	0.668/0.678	0.621/0.392	0.472/0.322	0.465/0.379	0.733/0.546	0.652/0.752
Straight+Circling	0.359/0.468	0.573/0.523	0.503/0.474	0.507/0.414	0.571/0.684	0.596/0.410
Straight+Bending	0.453/0.427	0.462/0.574	0.507/0.746	0.435/0.510	0.646/0.823	0.783/0.763
Circling+Bending	0.600/0.581	0.469/0.313	0.766/0.480	0.389/0.283	0.563/0.522	0.738/0.685
Circling+Straight	0.470/0.434	0.388/0.661	0.595/0.377	0.348/0.429	0.664/0.312	0.621/0.891
Bending+Straight	0.327/0.374	0.409/0.582	0.769/0.379	0.375/0.534	0.609/0.927	0.500/0.316
Bending+Circling	0.674/0.419	0.528/0.251	0.525/0.879	0.442/0.387	0.715/0.539	0.688/0.819
Overall accuracy (%)	49.18	48.33	53.69	41.64	64.80	68.47

Note. This table shows the cluster result of synthetic data set. The two numbers in each cell stand for precision/recall accordingly. EDR = edit distance on real sequence; LCSS = longest common subsequences; DTW = dynamic time warping; GRU = gated recurrent unit; LSTM = long short term memory.

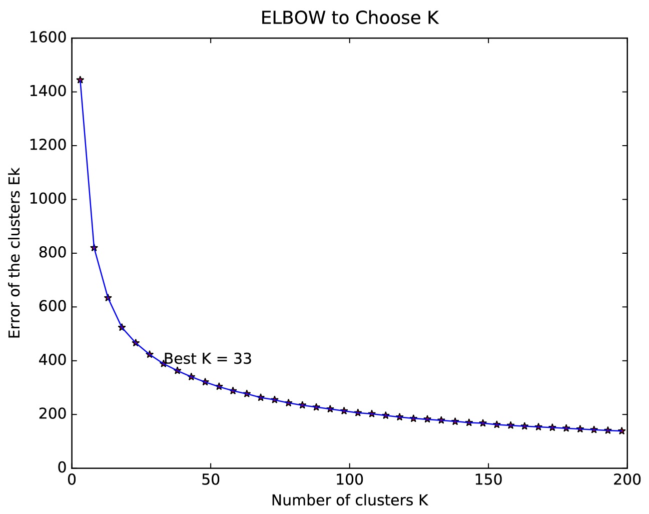

FIGURE 13 ELBOW method to choose K. E_k with suitable K value should be the elbow point in this figure. Here, we choose $K = 33$

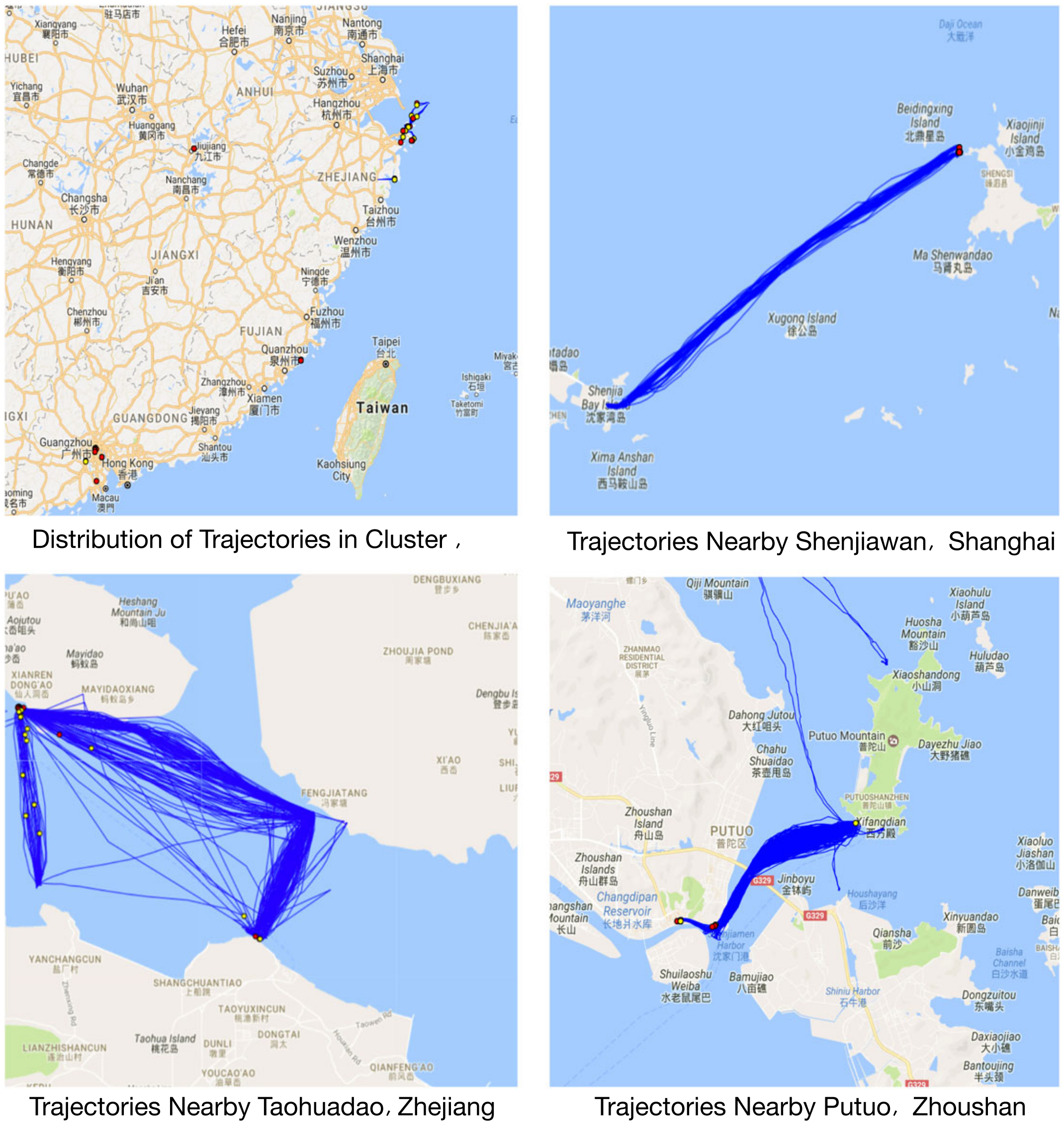
Distribution of Trajectories in Cluster，
Trajectories Nearby Shenjiawan，Shanghai
Trajectories Nearby Taohuadao，Zhejiang
Trajectories Nearby Putuo，Zhoushan

FIGURE 14 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 the trajectories in this cluster are short round trips between tourist cities and generated by passenger ships

in *MSE* (Equation 15). We choose 40 trajectories from both synthetic and real-life data randomly and measure the average training error in each epoch.

The results are shown in Figures 10, 11, and 12. In general, the training error first decreases and then becomes stable while m and n increase. The learning rate α plays an important role as well. To achieve a trade-off between effectiveness and efficiency, we set $m = 250$, $n = 400$, and $\alpha = 0.00001$ for LSTM; and $m = 100$, $n = 800$, and $\alpha = 0.00001$ for GRU.

5.3 | Results on synthetic data set

For the synthetic data, we set the sliding window to 600 s and the offset of the window to 300 s. 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 m. The clustering performance of different methods is shown in Table 1.

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 behaviours are clustered together, even if the similarity occurs in different regions and time periods. Our method has improved the accuracy by more than 10% than the other methods used for comparison.

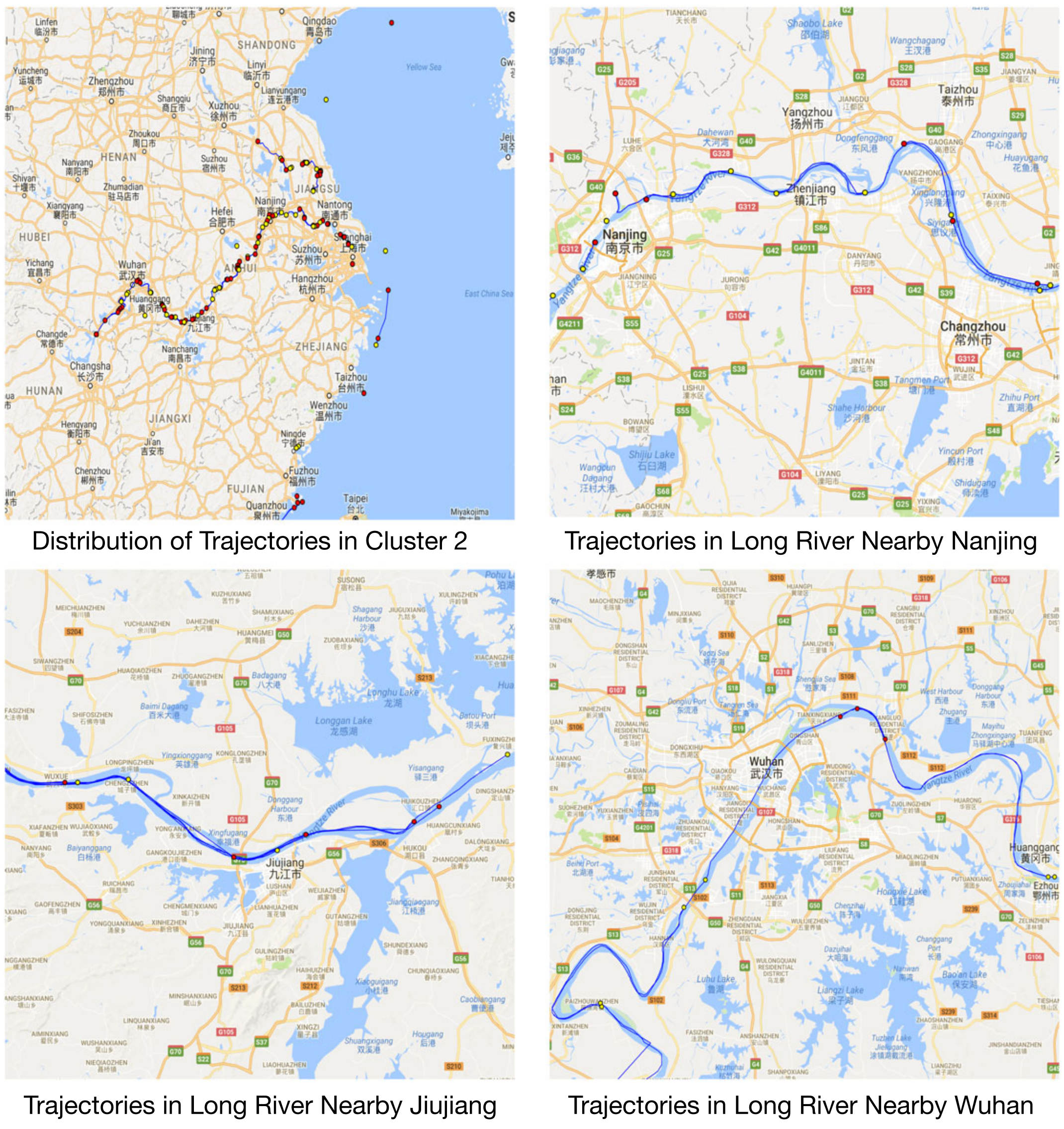
Distribution of Trajectories in Cluster 2
Trajectories in Long River Nearby Nanjing
Trajectories in Long River Nearby Jiujiang
Trajectories in Long River Nearby Wuhan

FIGURE 15 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 the 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

TABLE 2 Vessel type clustering results

	Passenger	Fishing	Cargo	Oil
Total number	50	50	50	50
Precision	46/53=0.87	38/44=0.86	23/37=0.62	50/66=0.76
Recall	46/50=0.92	38/50=0.76	23/50=0.46	50/50=1.0
Overall accuracy: (46+38+23+50)/200 = 0.785				

5.4 | Results on vessel motion data set

We perform two tasks on this data set. The first one is the standard trajectory clustering task, where we utilize our framework to generate clusters that have similar moving behaviour and then analyse the meaning of trajectories in them. The second one is vessel type analysis in which we examine whether the vessels owning the same type are grouped into the same cluster or not and measure the accuracies.

Trajectory clustering task: Utilizing our framework, trajectory moving behaviour vectors are generated. The parameters used in this procedure are the same as synthetic data set. Observing LSTM is better than GRU, we choose LSTM model in this task. Also, We use K-means to generate the clusters for the Z set.

As the number of ground-truth clusters in the real data is unknown, the procedure of choosing the value of K is conducted as follows. We set the value of K from 3 to 100 in 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 13 that depicts the K value is corresponding to the elbow point.

As shown in Figure 13, we choose $K = 33$, extracting 33 clusters for the 4,700 trajectories. Some of the cluster results are shown in Figures 14 and 15. The blue lines stand for the trajectories, whereas the yellow points stand for the start point; and the red points stand for the end point. The first cluster containing 117 trajectories is depicted in Figure 14. As shown, most of trajectories are distributed in tourist city. Besides, most of them are the short round trips. We find that the trajectories in this cluster are mostly generated from short passenger ship. The cluster in Figure 15 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 found that most of them were 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 analysing the typical trajectories in them.

Vessel type analysis task: Previous studies have shown that different vessel types have different behaviour patterns (de Souza, Boerder, Matwin, & Worm, 2016; Mazzarella, Vespe, Damalas, & Osio, 2014). In this task, we try to recognize the vessel type by utilizing the trajectory clustering.

We take the trajectory moving behaviour vectors of a vessel as the input of encoder and minimize the mean squared error between encoder input and decoder output. Subsequently, we obtained the moving behaviour vector of the vessel. Based on these vessel moving behaviour 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 78.5%. Especially, the precision/recall for the oil ship and the passenger ship are 0.76/1.0 and 0.87/0.92, respectively. However, the result of cargo ship is only 0.62/0.46. 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, wet cargo ship, and roll-on-roll-off ship. These different types make a great difference in the moving behaviour patterns. However, if such subtype information is available in the training process, the cluster performance is expected to be better.

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

6 | CONCLUSION

In this paper, we proposed a novel framework for trajectory clustering with similar movement patterns. A moving behaviour feature extraction algorithm was proposed to extract moving behaviour features that captured space- and time-invariant characteristics of trajectories. Then, a sequence-to-sequence auto-encoder is utilized to generate a deep representation of moving behaviour sequence and address the spatio-temporal shifts problem. We have demonstrated the effectiveness of our framework on both synthetic and real data sets. 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.

ACKNOWLEDGEMENTS

This work is supported by research Grant 61472403 and 61303243 from National Natural Science Foundation of China (NSFC). An earlier version of this paper (Yao, Zhang, Zhu, Huang, & Bi, 2017) was presented at the International Joint Conference on IJCNN 2017.

ORCID

Di Yao http://orcid.org/0000-0003-1778-8319

REFERENCES

Alahi, A., Vignesh Ramanathan, K. G., Robicquet, A., Li, F.-F., & Savarese, S. (2016). Social LSTM: Human trajectory prediction in crowded spaces, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, pp. 961–971.

Bengio, Y., Courville, A. C., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Bengio, Y., Simard, P. Y., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Besse, P. C., Guillouet, B., Loubes, J. M., & Royer, F. (2016). Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, *17*(11), 3306–3317.

Chen, L., Özsu, M. T., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, ACM, New York, NY, USA, pp. 491–502.

Chen, Q., Song, X., Yamada, H., & Shibasaki, R. (2016). Learning deep representation from big and heterogeneous data for traffic accident inference. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, AAAI Press, Phoenix, Arizona, pp. 338–344.

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *EMNLP*, Association for Computational Linguistics, Doha, Qatar, pp. 103–111.

Chung, Y.-A., Wu, C.-C., Shen, C.-H., & Lee, H.-Y. (2016). Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks. In *Proc. Interspeech*, San Francisco.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*, Montréal, pp. 3079–3087.

de Souza, E. N, Boerder, K., Matwin, S., & Worm, B. (2016). Improving fishing pattern detection from satellite ais using data mining and machine learning. *PloS One*, *11*(7), e0158248.

Harati-Mokhtari, A., Wall, A., Brooks, P., & Wang, J. (2007). Automatic identification system (ais): Data reliability and human error implications. *Journal of Navigation*, *60*(03), 373–389.

Higgs, B., & Abbas, M. M. (2015). Segmentation and clustering of car-following behavior: Recognition of driving patterns. *IEEE Transactions on Intelligent Transportation Systems*, *16*(1), 81–90.

Hung, C.-C., Peng, W.-C., & Lee, W.-C. (2015). Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *VLDB Journal*, *24*(2), 169–192.

Keskin, C., Cemgil, A. T., & Akarun, L. (2011). DTW based clustering to improve hand gesture recognition, *HBU*, Lecture Notes in Computer Science, Vol. *7065*: pp. (72–81), Springer, Amsterdam, The Netherlands.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*(1-3), 1–6.

Lee, J.-G., Han, J., & Whang, K.-Y. (2007). Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, ACM, New York, NY, USA, pp. 593–604.

Li, Z., Lee, J.-G., Li, X., & Han, J. (2010). Incremental clustering for trajectories, *DASFAA (2)*, Lecture Notes in Computer Science, Vol. *5982* (pp. 32–46). Springer, Berlin, Heidelberg.

Liu, S., Ni, L. M., & Krishnan, R. (2014). Fraud detection from taxis' driving behaviors. *IEEE Transactions on Vehicular Technology*, *63*(1), 464–472.

Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures, *ICDM* (pp. 911–916) IEEE Computer Society, Sydney, NSW, Australia.

Mazzarella, F., Vespe, M., Damalas, D., & Osio, G. (2014). Discovering vessel activities at sea using AIS data: Mapping of fishing footprints. In *17th International Conference on Information Fusion (FUSION)*, Salamanca, pp. 1–7.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., ..., Ward, R. K. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, *24*(4), 694–707.

Sas, C., O'Hare, G. M. P., & Reilly, R. (2005). Virtual environment trajectory analysis: A basis for navigational assistance and scene adaptivity. *Future Generation Computer Systems*, *21*(7), 1157–1166.

Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. In Francis Bach and David Blei (Eds.) *Proceedings of the 32nd International Conference on International Conference on Machine Learning - (ICML'15)*, Vol. 37. JMLR.org, pp. 843–852.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - (NIPS'14)*, Vol. 2. MIT Press, Cambridge, MA, USA, pp. 3104–3112.

Tang, W., Pi, D., & He, Y. (2016). A density-based clustering algorithm with sampling for travel behavior analysis *IDEAL*, Lecture Notes in Computer Science, Vol. *9937*. (pp. 231–239). Springer, Cham: Springer.

Yao, D., Zhang, C., Zhu, Z., Huang, J., & Bi, J. (2017). Trajectory clustering via deep representation learning. In *2017 International Joint Conference on Neural Networks, IJCNN 2016, Anchorage, AK, US, May 13-19, 2017*, pp. 4097–4104.

Yuan, G., Sun, P., Zhao, J., Li, D., & Wang, C. (2017). A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, *47*(1), 123–144.

Yuan, Q., Zhang, W., Zhang, C., Geng, X., Cong, G., & Han, J. (2017). PRED: Periodic region detection for mobility modeling of social media users. In *WSDM*, ACM, pp. 263–272.

Zhang, C., Han, J., Shou, L., Lu, J., & Porta, T. F. L. (2014). Splitter: Mining fine-grained sequential patterns in semantic trajectories. *PVLDB*, *7*(9), 769–780.

Zhang, C., Zhang, K., Yuan, Q., Zhang, L., Hanratty, T., & Han, J. (2016). Gmove: Group-level mobility modeling using geo-tagged social media. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1305–1314.

Zheng, Y. (2015). Trajectory data mining: An overview. *ACM TIST*, *6*(3), 29:1–29:41.

Di Yao received his BS degree in software engineering from Northeastern University in 2013. He is currently pursuing Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences under the supervision of Prof. Jingping Bi. His research interest lies on spatio-temporal data mining and deep learning.

Chao Zhang received his BS, MS degree in computer science from Zhejiang University in 2010 and 2013. He is currently pursuing Ph.D. degree in the Computer Science Department of UIUC under the supervision of Prof. Jiawei Han. His research interests include spatio-temporal data mining, indexing techniques, and web data mining.

Zhihua Zhu received his BE degree in computer science and technology from Nanjing University of Aeronautics and Astronautics in 2015. He is currently pursuing Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences under the supervision of Prof. Jingping Bi.

Qin Hu is currently a graduate student at College of Information Science and Technology of Beijing Normal University. Her current major is Computer Application Technology. Her major interests are big data, data mining, and mobile social network.

Zheng Wang received his BE degree with 1st class honours in software engineering from University of Sydney in 2011. He is currently pursuing Ph.D. degree in the School of Information Technologies, University of Sydney, under the supervision of A/Prof. Irena Koprinska. His research interests include time series prediction and feature selection.

Jianhui Huang received the Ph.D. degree in computer science from Xi'an Jiaotong University, China, in 2009. He is currently a Research Assistant at the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include the mobility management, routing, and cloud computing.

Jingping Bi received her Ph.D. in 2002 from the Institute of Computing Technology, Chinese Academy of Sciences. She is currently a full professor at the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include network measurement, routing, virtualization, SDN, and bigdata. She is an IEEE Member.

How to cite this article: Yao D, Zhang C, Zhu Z, et al. Learning deep representation for trajectory clustering. *Expert Systems*. 2018; 35:e12252. https://doi.org/10.1111/exsy.12252