Journal of Information Hiding and Multimedia Signal Processing ©2019 ISSN 2073-4212
Ubiquitous International Volume **10**, Number **2**, March **2019**

A Second-Order HMM Trajectory Prediction Method based on the Spark Platform

Xing Wang[1,2,*], Xinhua Jiang[1,3], Yi Wu[2] and Mingwei Lin[2]

[1]College of Information Science and Engineering,
Central South University, Changsha 410075, China
Corresponding author: wangxing@fjnu.edu.cn

[2]College of Mathematics and Informatics,
Fujian Normal University, Fuzhou 350117, China
linmwcs@163.com, csewx@163.com

[3]Fujian Key Laboratory of Automotive Electronic and Electrical Drive Technology,
Fujian University of Technology, Fuzhou 350117, China
xhjiang@fjut.edu.cn

Received November 2018; revised January 2019

Abstract. *A large quantity of spatiotemporal trajectory data is generated through the application of GPS technology and LBS. Faced with massive quantities of trajectory data, it is of great significance to effectively mine patterns and rules from that data. Statistical model-based methods, such as MM and HMM models, are widely used for this purpose. Substantial research has been conducted based on MM and related improved methods. Compared with the MM model, the HMM model can more fully statistically characterize data sets. Considering the problems that come with a large data application environment, and the impact on prediction accuracy, we propose a second-order HMM distributed trajectory prediction method (HMM2) based on the Spark platform. The HMM2 clusters the historical trajectories and then extracts the clusters, which are the hidden states of the model, and divides the road network space into a grid in which the grid cells are the observation states of the model. The HMM2 extracts the hidden sequences and observation sequences of historical trajectory in time and then uses the BW learning algorithm to train observation sequences to estimate the parameters of the model. For a given prediction sequence, the next observation state with the largest probability value is calculated by using a forward or backward probability algorithm, which is the result of prediction. Experimental results show that the proposed method has a good prediction effect and has a higher speedup ratio in a distributed environment than in stand-alone mode. Experiments on different datasets show that the proposed method is robust.*
Keywords: Hidden Markov Model, Trajectory Prediction, Distributed, Spark

1. **Introduction.** Due to the rapid popularization and widespread application of global positioning technology, equipment embedded with GPS sensors, including cars, smartphones, smart wristbands and shared bikes, is increasingly common. In that process, large quantities of trajectory data, such as vehicle driving paths, pedestrian traveling records and animal migration tracks, are constantly generated. Therefore, it is very important to effectively mine patterns and rules from trajectory data [1, 2]. Location prediction is the basis for providing active information services [3, 4]. For humanized and intelligent location service demands, we predict the movement trends of moving objects by mining the patterns and rules of movement trajectory [5]. There are many methods and strategies for location prediction, such as the discovery method based on frequent patterns [6, 7],

the trajectory similarity strategy [8, 9] and the statistical method based on the Markov model [10], hidden Markov model [11], etc.

Mao *et al.* [12] used a fixed-order Markov model for trajectory prediction. Nagata *et al.* [13] proposed a variable-order Markov model (VLMM) prediction method to solve the limitation of fixed-order Markov model prediction and the high complexity of high model order. For insufficient data sets, the low coverage problem of the models was considered, and a kernel smoothing variable-order Markov model (KVLMM) was proposed in the literature [14], which constructed the model in linear time and complex space and predicted the next location adaptively with appropriate order length. Compared with the Markov model, the HMM model is more effective in characterizing the statistical characteristics of sample data and is widely used in statistical modeling [15]. For example, the position prediction framework proposed in the literature [16] can realize a personalized destination prediction system. The method based on the k-nearest neighbor (kNN) and the decision tree combination method are used for location recognition, and the HMM model is used to train and predict the user destinations. Wu *et al.* [17] used the hidden Markov model to analyze the driving patterns of vehicles to predict the driving possibilities of vehicles and to change directions in a timely manner. Qiao *et al.* [18] conducted trajectory prediction based on the hidden Markov model. Their model used the clustering algorithm to partition and segment original trajectory data to reduce the quantity of HMM states. An adaptive parameter selection algorithm was proposed in the literature [19] to capture the dynamic changes of objects in real scenarios. In addition, the algorithm also preprocesses the trajectory segmentation to improve the prediction efficiency. However, relevant studies fail to consider trajectory prediction problems under large-scale data sets. How to address massive quantities of trajectory data quickly is also an important question.

Herein, we design a second-order HMM model trajectory prediction method (HMM2) for large-scale data sets based on a Spark platform to meet the demands of trajectory prediction.

First, we define and explain the principle of the second-order hidden Markov model and then design the trajectory prediction framework based on the Spark platform, introducing the three main processes of the prediction framework: data preprocessing method, model learning and training, and model prediction. Finally, we performed comparative experiments.

2. Related definitions and principles.

The second-order HMM model is used to carry out trajectory training and prediction. It includes probability calculation of observation sequences and learning estimation of model parameters. Relevant definitions and algorithms are as follows:

Definition 1 HMM2 uses $\lambda = (\pi, A_1, A_2, B_1, B_2)$ to describe the model. Suppose $X(n)$ represents the trajectory hidden state sequence, $O(n)$ represents the observation state sequence as follows:

$\pi = \{\pi_i\}$: Probability distribution of trajectory initial state; $\pi_i = P(X_1 = S_i)$ refers to the probability of selecting a state during the initialization.

$A_1 = \{A_{ij}\}$: First-order hidden state transition probability matrix, $A_{ij} = P(X_{t+1} = S_j \,|X_t = S_i)$ means that the probability under the hidden state S_i at time t o'clock and the probability under the hidden state S_j at time $t + 1$.

$A_2 = \{A_{ijk}\}$: Second-order hidden state transition probability matrix; $A_{ijk} = P(X_{t+1} = S_k | X_t = S_j,\ X_{t-1} = S_i)$ means that the probability under the hidden state S_i at time $t - 1$ and the probability under the hidden state S_k at time $t + 1$.

$B_1 = \{B_{kl}\}$: First-order lower trajectory observation state and hidden transition probability matrix, also known as the confusion matrix, where $B_{kl} = P(O_t = O_l | X_t = S_k)$

describes the probability of obtaining the observation state O_l from the hidden state S_k at time t.

$B_2 = \{B_{jkl}\}$: The second-order observation state and hidden transition probability matrix, where $B_{jkl} = P(O_t = O_l | X_t = S_k,\ X_{t-1} = S_j)$ describes the probability of the observation state O_l obtained from the hidden state S_k at a certain time t and hidden state S_j at time $t-1$.

Definition 2 (Forward Probability) Under the given second-order HMM model, some observation sequences by time t are defined as $O_1, O_2, ..., O_t$. Its probability under the hidden state S_i at time $t-1$ and under the hidden state S_j at time t is regarded as the forward probability. These are written as $\alpha_t(i,j) = P(O_1, O_2, ..., O_t,\ X_{t-1} = S_i,\ X_t = S_j|\lambda)$, which is consistent with the first-order HMM forward algorithm. The $\alpha_t(i,j)$ and sequential probability $P(O|\lambda)$ can be calculated by recursion.

$$P\left(O|\lambda\right) = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_T\left(i,j\right) \tag{1}$$

Definition 3 (Backward Probability) Backward probability is similar to the definition of the forward probability. For the second-order HMM model λ and S_i at time $t-1$ and S_j at time t, the probability of some observation sequences of $O_{t+1}, O_{t+2}, ..., O_T$ from time $t+1$ to time t is referred to as the backward probability, which is written as follows:

$$\beta_t\left(i,j\right) = P(O_{t+1}, O_{t+2}, \ldots, O_T,\ X_{t-1} = S_i, X_t = S_j|\lambda) \tag{2}$$

The probability of an observation sequence can also be expressed by forward probability and backward probability as follows:

$$\mathrm{P}\left(O|\lambda\right) = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t\left(i,j\right)\cdot\beta_t\left(i,j\right) \tag{3}$$

Setting the initial parameter of the model as $\lambda = (\pi, A_1, A_2, B_1, B_2)$ and the observation sequence $O = O_1, O_2, ..., O_t$, then

$$\begin{aligned}\xi_t(i,j,k) &= P(X_{t-1} = S_i,\ X_t = S_j,\ X_{t+1} = S_k|O,\lambda)\\ &= \frac{\alpha_t(i,j)\cdot A_{ijk}\cdot B_{ij}(O_{t+1})\cdot\beta_{t+1}(j,k)}{P(O|\lambda)}\end{aligned} \tag{4}$$

$$\begin{aligned}\gamma_t(i,j) &= P(X_{t-1} = S_i,\ X_t = S_j\,|O,\lambda)\\ &= \frac{\alpha_t(i,j)\cdot\beta_t(i,j)}{P(O|\lambda)}\end{aligned} \tag{5}$$

Among the above equations, $\xi_t(i,j,k)$ represents the probability for state transition at time t, while $\gamma_t(i,j) = \sum_{k=1}^{N}\xi_t(i,j,k)$ represents the probability under the state S_i at time $t-1$ and under the state S_j at time t. According to the maximum likelihood estimation method, the Baum-Welch learning formula for the second-order HMM model can be expressed as follows:

$$\pi_i = \sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{w=1}^{W}\xi_{\mathrm{t}=2}^{w}\left(i,j,k\right) \tag{6}$$

$$A_{ij} = \frac{\sum_{k=1}^{N}\sum_{w=1}^{W}\xi_{t=2}^{w}\left(i,j,k\right)}{\sum_{j=1}^{N}\sum_{w=1}^{W}\sum_{k=1}^{N}\xi_{\mathrm{t}=2}^{w}\left(i,j,k\right)} \tag{7}$$

$$A_{ijk} = \frac{\sum_{t=2}^{T-1} \sum_{w=1}^{W} \xi_t^w (i,j,k)}{\sum_{t=2}^{T-1} \sum_{w=1}^{W} \sum_{k=1}^{N} \xi_{\mathrm{t}}^w (i,j,k)} \tag{8}$$

$$B_{kl} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{w=1}^{W} \xi_{t=2}^w (i,j,k) \cdot \delta o_1^w = v_l}{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{w=1}^{W} \xi_{\mathrm{t}=2}^w (i,j,k)} \tag{9}$$

$$B_{jkl} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t-2,Ot=O1}^{T-1} \sum_{w=1}^{W} \xi_t^w (i,j,k)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=2}^{T-1} \sum_{w=1}^{W} \xi_{\mathrm{t}}^w (i,j,k)} \tag{10}$$

3. **A Second-order HMM trajectory prediction framework based on Spark.** Figure 1 shows the framework of the user trajectory prediction algorithm based on the Spark platform. It mainly consists of three parts: trajectory data preprocessing, HMM2 model training based on Spark, and prediction processing.

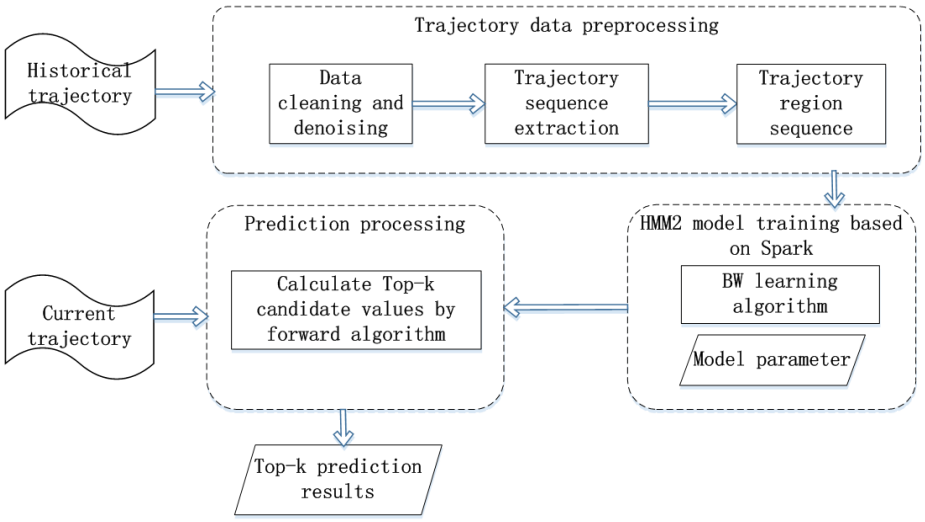

FIGURE 1. Trajectory prediction framework based on the Spark platform

(1) Trajectory data preprocessing

It is necessary to preprocess historical travel trajectories, such as filtering the noise data, abstracting processing of trajectories, and extracting a sequence of historical travel areas of users.

1) Trajectory data denoising

Due to the lack of GPS positioning signals, data drift, etc., there is a large amount of data noise. To filter the noise data out of the original trajectory data, the trajectory data points with the same spatial density characteristics are divided into a set by the DBSCAN clustering algorithm so that the noisy data points can be effectively removed.

2) Trajectory sequence extraction

During the process of removing noisy data points, the DBSCAN algorithm is used to cluster the historical trajectory data and obtain the clustering clusters, which are the

hidden state set of the hidden Markov model. The historical trajectory sequence can be transformed into a cluster sequence according to time. Additionally, the trajectory sequence can be extracted according to the grid division. The road network space is divided into grids, which are the observation state sets of the hidden Markov model, and the sequence of grid cells is the observation sequence. Hidden states are shown in Figure 2, S1, S2, S3 and observation states are shown in Figure g1, g2, g3, etc.

The division of the grid size determines the number of observation states of the hidden Markov model. A small partition will lead to a cluster covering multiple grids; that is, one hidden state is associated with multiple observation states. A large partition will lead to the problem of one grid cell containing multiple clusters; that is, one observation state is associated with multiple hidden states. In addition, due to the nonuniform distribution characteristics of the trajectory data, the parameters of clustering need to be reasonably set.

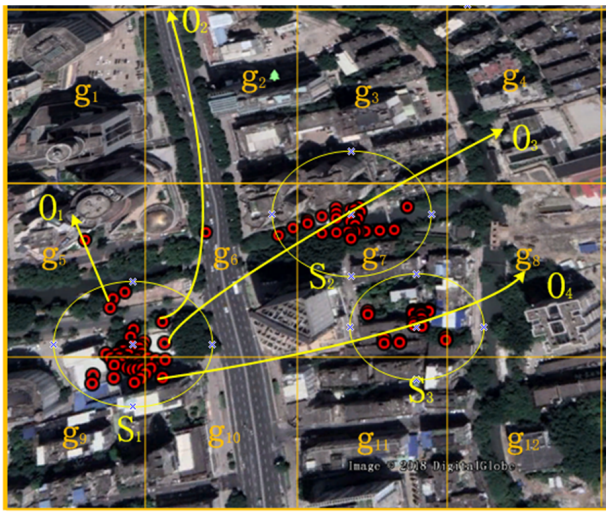

FIGURE 2. Trajectory sequence extraction

(2) HMM2 model training based on Spark

Spark is a general parallel framework based on the Hadoop platform. It has the advantages of Hadoop MapReduce. However, unlike MapReduce's repeated reading and writing HDFS, it is a memory-based computing model that can store intermediate results in memory, can greatly improve the efficient processing ability of the data stream and has great advantages in performance. It can be applied to machine learning and other algorithms that require iterative calculation. The computing framework of the distributed learning algorithm for the second-order HMM model in a large data environment is shown in Figure 3.

The pseudo code of the learning algorithm based on Spark is as follows:

Algorithm 1 HMM2 distributed learning training

Input: path, nState, nEsym, observe, nITER, partitionMaxCnt
path: training data set HDFS storage path
nState: the number of hidden states
nEsym: number of observation states
observe: all possible observations
nITER: algorithm iterations

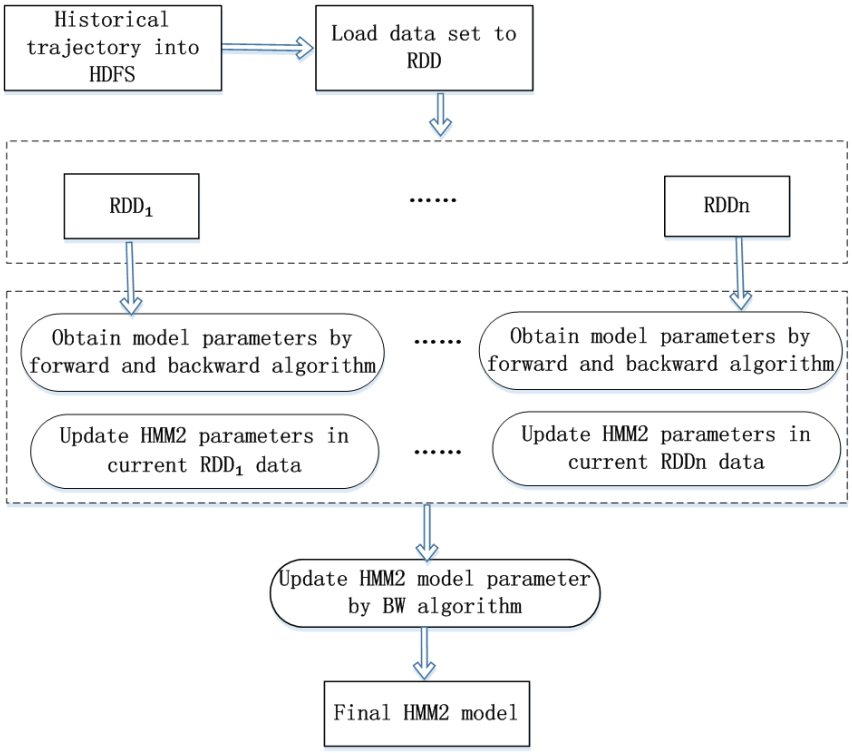

FIGURE 3. HMM2 distributed learning algorithm

partitionMaxCnt: the maximum number of partitions
Output: model parameters $\lambda = (\pi, A_1, A_2, B_1, B_2)$
(1) lines ← readFromHdfs(path)
(2) rdd ← lines.map{ x=> distributeIDToLineData(partitionMaxCnt) }
(3) matA1,matA2,Pi,matB1,matB2 ← initByRandom(nSate, nEsym)
(4) Xi2 ← initMatrixXi2ByZero(lines.size, nState, nEsym)
(5) for all in nITER do
(6) logP, tmpB1,tmpB2,tmpA1,tmpA2,tmpPi ← initMatrixByZero(nState, nEsym)
(7) rdd.map{ x=> logP, alphas,betas ← doForwardBackward(Xi2)
(8) update(Xi2, tmpB, tmpA1, tmpA2, tmpPi) }
(9) matA1, matA2, Pi, matB1, matB2 ← updateParam(tmpB,tmpA1,tmpA2,Xi2)
(10) println(logP)
(11) return hmm(Pi, matA1, matA2, matB1, matB2)

Line 1 reads the training data set from the HDFS file management system.

Line 2 divides the RDDs, and the ID of each partition's data set is marked to facilitate subsequent statistical processing.

Lines 3 and 4 initialize the relevant parameters.

Lines 5-11 perform the algorithm iterative process. First, the local variables are initialized to accelerate the subsequent update of the HMM parameters.

Line 7 performs forward and backward algorithm processing on each of the data sets in the partition to obtain alphas and betas matrix information.

Line 9 is used to speed up the update of HMM parameters.

Line 11 generates the basic HMM structure. The current algorithm further enhances the execution speed of the algorithm by consuming space in the above Spark framework.

(3) Prediction processing

Given the current trajectory, the trained HMM2 model is used to predict the next grid cell. The current trajectory sequence is transformed into a grid sequence, which is input into the trained HMM2 model, and the grid sequences with the top k values are calculated by the forward probability algorithm. The pseudo code of the prediction algorithm is as follows:

Algorithm 2 Trajectory Prediction Algorithm

```
Input: pts, Roads, Hmm, K
       pts: set of trajectory points to be processed
       Roads: grids set
       Hmm: Hidden Markov Model
       K: the number of the maximum prediction grid cell
Output: Whether the trajectory prediction is correct
(1) seqDatas ← convertTrajectory(pts, roads)
(2) if (seqDatas.size() < 3) // check if the sequence length is > 3
(3)      return false;
(4) (datasDeal, dataCheck) ← splitIDs(seqDatas)
(5) roadsWithPro ← new List
(6) for all road in roads do
(7)      pro ← forward(hmm, idsDeal + road)
(8) roadsWithPro.add(road, pro)
(9) roadPredicats ← sort(roadsWithPro).getTop(K)
(10) if roadPredicats.Contain(idCheck)
(11)      return true;
(12) return false
```

In lines 1-3, feature extraction processing on the current travel trajectory of the user to obtain a sequence of grid cells of the user is performed.

In line 4, a new list is created for storing each set of feature information and the last set of feature information as the probability value of the next predicted link of the current sequence.

In line 7, the forward algorithm of the second-order HMM model is used to calculate the probability of occurrence of this sequence, and the results are saved to the list.

In line 9, the list is sorted according to the stored probability value, and the top k with the larger probability value is taken as the prediction grid cell of the current trajectory.

In lines 10-12, whether there is a last grid cell ID of the current trajectory sequence in the top k prediction grids is checked.

4. **Experimental results and analysis.** To verify the prediction effect of the proposed method, we implemented three prediction methods: a prediction method based on a standard order hidden Markov model (HMM), a prediction method based on a second-order hidden Markov model (HMM2) and a prediction method based on a variable-order Markov model based on kernel smoothing (KVLMM). Experiments are conducted from the aspects of grid division, cluster size setting, the ratio of observation and hidden state numbers as well as trajectory sequence size.

(1) Experimental configuration

We implement the algorithms with Java and conduct the experiments on four PCs with an Intel(R) Core(TM), i5-3470 CPU with 2.8 GHz (4 CPUs) and 4 GB main memory running the Windows 10 operating system. One PC is a master computer, and the three

PCs are slaver computers. Experiments are deployed on a Spark platform in distributed mode.

Experimental data set 1: Geolife data set. This data set is collected from 182 users in Microsoft's Asian Research Institute Geolife project over five years (from April 2007 to August 2012). These data consist of a series of time-stamped points, each including latitude, longitude and altitude information. Moreover, the data set contains 17,621 tracks with a total distance of 1,292,951 km and a total duration of 50,176 hours.

Experimental data set 2: floating car data (FCD) in Fuzhou, Fujian Province. This data set consists of floating vehicle trajectory data from May 2016. It covers the geographic area of approximately 1430 km2 with a longitude range of [119.113, 119.684], a latitude range of [25.904, 26.155].

(2) Experimental results and analysis

1) The effect of grid division on prediction accuracy

Here, we first discuss the effect of grid size settings on prediction accuracy. To verify the effect of grid division on accuracy, observation sequences are extracted in different grid sizes, and experimental results in data set 2 are shown in Figure 4.

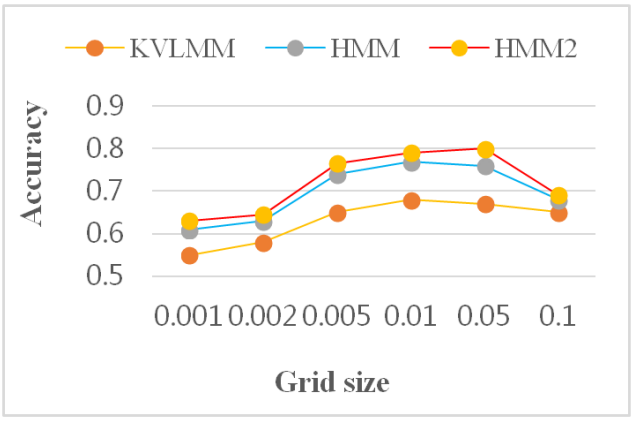

FIGURE 4. Effect of grid division on prediction accuracy (top 1 prediction)

A grid size of 0.001 represents 10 meters. The prediction accuracy varies with the grid size. Excessively small grid division leads to dense extraction of the trajectory sequence, too much sequence state and low prediction accuracy. Excessively large grid division leads to a sparse sequence of trajectory extraction, and prediction accuracy begins to decrease. The experimental results show that when the grid size is approximately 50 100 meters, the prediction accuracy is optimal.

2) The effect of cluster size on prediction accuracy

In road networks, the distribution of the trajectory data is nonuniform. When using the DBSCAN algorithm for clustering, different cluster parameters will lead to differences in the number of clusters, which affect the extraction of trajectory hidden state sequences. Experiments are carried out in data set 1 and data set 2. The results are shown in Figure 5 and Figure 6.

The trajectory data in data set 1 are mostly user travel data, with a low sampling frequency, a large spatiotemporal span and relatively sparse data points. Therefore, when the clustered parameter settings are Minpts = 50∼100 and Eps = 50 meters, the effect is best. In data set 2, the trajectory data are mainly taxi data, with high and dense sampling frequencies. When the parameter settings are Minpts = 300∼500 and Eps = 50 meters, the effect is best.

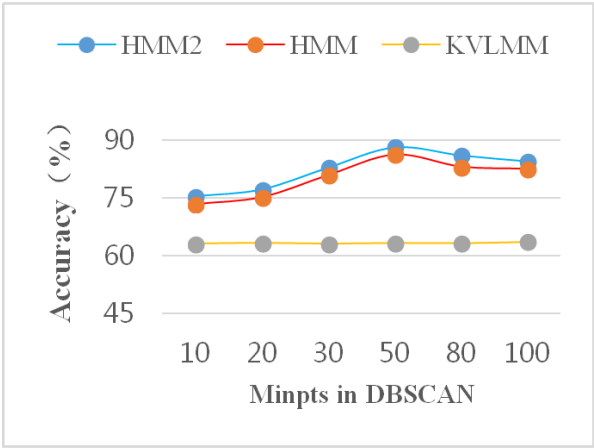

FIGURE 5. Geolife data set experiments

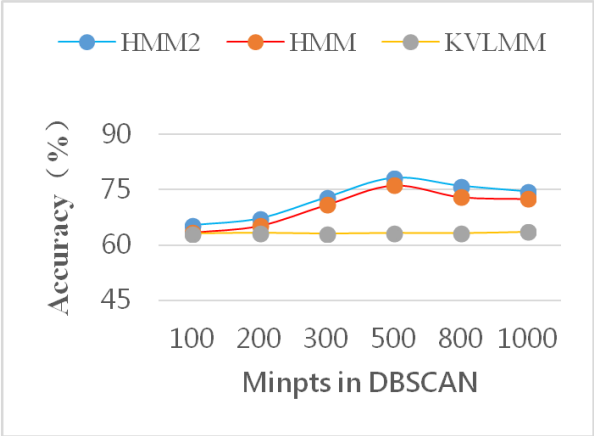

FIGURE 6. FCD data set experiments

3) Effect of different explicit and implicit numbers on comparison precision

The appearance of the observation state sequence of the hidden Markov model contains the corresponding hidden state sequence. There is a specific association between the hidden state sequence and the observation state sequence. We obtain the model's hidden state and observation state numbers by adjusting the size of clusters and grids. Experiments are conducted on the Geolife data sets using 7-day travel data from multiple users. The average experimental results are shown in Figure 7.

The grid size and cluster affect the number of observation states and the number of hidden states, respectively. When the ratio of observation states to hidden states is too small, the number of observation states is obviously less than the number of hidden states, and the prediction accuracy is low. As the ratio increases, the number of hidden states and the number of observation states are relatively balanced, and the prediction accuracy is gradually improved. The prediction effect is optimal when the ratio is 0.6∼0.8. As the ratio increases further, the number of hidden states and the number of observation states tend to be unbalanced, and the prediction accuracy decreases.

4) Comparison of data sets in different sizes

According to the above results, to better distinguish the influence of data sets on the prediction accuracy, the ratio of observation state numbers and hidden state numbers in this experiment is 0.6. The experimental results are shown in Figure 8 and Figure 9.

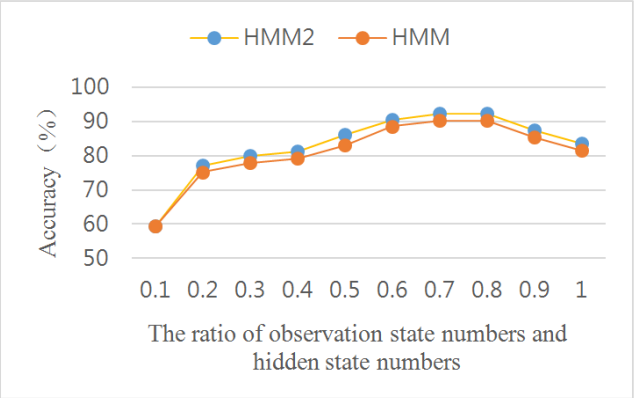

FIGURE 7. Prediction accuracy (top 3 predictions)

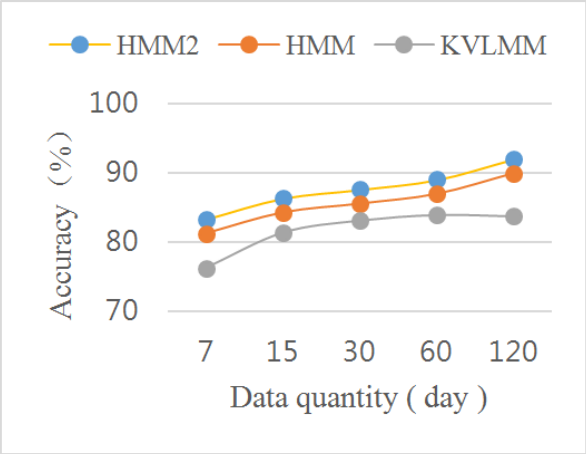

FIGURE 8. Geolife dataset (top 3 predictions)

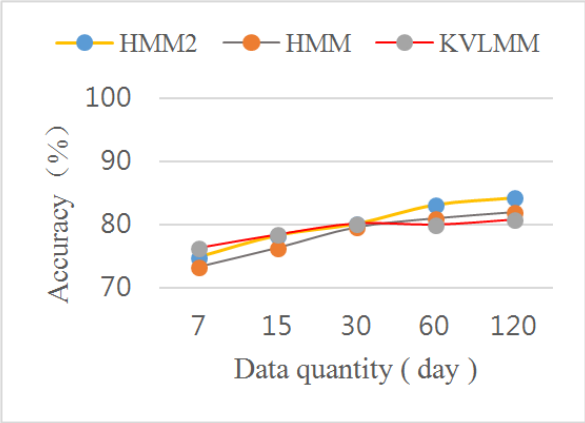

FIGURE 9. FCD dataset (top 3 predictions)

The experimental results show that when the amount of trajectory data increases, the prediction accuracy increases gradually. In the Geolife dataset, the prediction accuracy is stable at approximately 90% as shown in Figure 8. For the FCD dataset in Fuzhou, shown in Figure 9, the average accuracy rate is 82.7% due to the large variation in taxi

sample data, which reduces the accuracy relative to the Geolife data. With the increase in data quantity, the prediction accuracy of the KVLMM model is gradually improved, but the curve changes more smoothly, the learning ability of the HMM and HMM2 models is stronger, the prediction accuracy is obviously improved, and the effect is better than that of KVLMM.

5) Comparison of computational efficiency

We compare the computational efficiency of the learning algorithm in stand-alone and Spark distributed mode. The ratio of observation state numbers and hidden state numbers in this experiment is 0.4, and the iteration number of the learning algorithm is 100 times. The comparison of results is shown in Table 1 and Figure 10.

TABLE 1. Comparison of runtimes in stand-alone and Spark distributed mode

Size (sequence)	Stand-alone mode (second)	Distributed mode (second)
10	89.8	31.4
100	800.7	219.4
1000	5618.6	1824.8
10000	49512.3	16574.8
100000	518000	172000

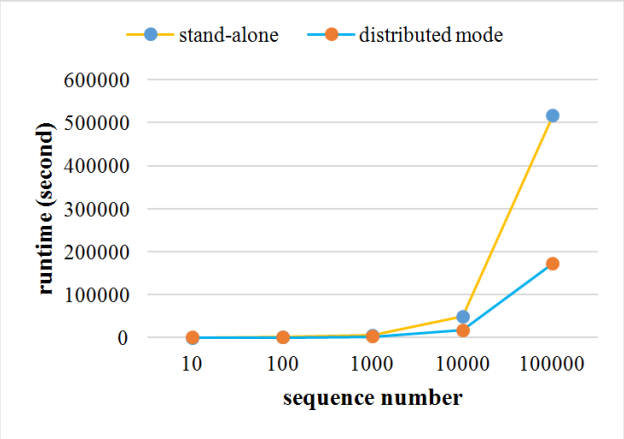

FIGURE 10. Comparison of runtimes in stand-alone and Spark distributed modes

When the trajectory sequence size is not large, the algorithm operating efficiency between the stand-alone mode and the distributed mode is not obviously different. However, when the scale sequence exceeds a certain number, the running time of the algorithm sharply increases in stand-alone mode. In distributed mode, the running time of the algorithm increases steadily. Comparison experiments between stand-alone and distributed mode show that the algorithm has a good speedup ratio.

5. **Conclusions.** The Markov model only describes the statistical characteristics of a single dimension in a data set but fails to describe the implicit relationship among multiple dimensions in a data set. In this paper, we propose a second-order hidden Markov trajectory prediction method based on the Spark platform. The formulas of the second-order hidden Markov model are given, and the trajectory prediction framework under the Spark platform is designed. The data preprocess, training algorithm, prediction algorithm and other specific processes are described. The experimental results show that the HMM

model is better than the MM model in characterizing the statistical characteristics of data sets. The proposed method has a good prediction effect and has a higher computational efficiency and a higher acceleration ratio on the Spark platform. Experiments on different data sets show that the proposed method is more robust than other tested methods. The next step is to introduce more dimensions to classify data sets or integrate multisource information to extract comprehensive features to further improve the prediction accuracy of this method.

Acknowledgment. This work is supported in part by the National Natural Science Foundation of China under Grant 61872086 and Grant 61502103, in part by the Fujian Natural Science Foundation under Grant 2016J05149.

REFERENCES

[1] X Huang, T Wang, S Li S, et al., Valuable group trajectory pattern mining directed by adaptable value measuring model, *Web-Age Information Management. Springer International Publishing*, pp. 380-391, 2016.

[2] M Qiu, D Pi, Mining frequent trajectory patterns in road network based on similar trajectory, *Intelligent Data Engineering and Automated Learning V IDEAL 2016. Springer International Publishing*, 2016.

[3] Z Dong, J Deng, X Jiang, et al.d RTMatch: real-time location prediction based on trajectory pattern matching, 2017.

[4] D Liao, G Sun, H Li, et al,. The framework and algorithm for preserving user trajectory while using location-based services in IoT-cloud systems, *Cluster Computing*, vol. 20, no. 2, pp. 1-15, 2017.

[5] Z Xia, Z Hu, J Luo, UPTP vehicle trajectory prediction based on user preference under complexity environment, *Wireless Personal Communications*, vol. 12 pp. 1-15, 2017.

[6] B Aydin, R Angryk, Spatiotemporal frequent pattern mining on solar data: current algorithms and future directions, *IEEE International Conference on Data Mining Workshop. IEEE*, pp. 575-581, 2016.

[7] O Obulesu, D A R M Reddy, M Mahendra, A comparative study of frequent and maximal periodic pattern mining algorithms in spatiotemporal databases, vol. 225, no. 1, pp. 012066, 2017.

[8] Z Wang, W Tang, D Pi, Trajectory similarity-based prediction with information fusion for remaining useful life, *International Conference on Intelligent Data Engineering and Automated Learning. Springer, Cham*, pp. 270-278, 2017.

[9] M Chen, X Yu, Y Liu, Mining object similarity for predicting next locations, *Journal of computer science and Technology (English version)*, vol. 31, no. 4, pp. 649-660, 2016.

[10] H Abdel-Fatao, J Li, J Liu, STMM: semantic and temporal-aware markov chain model for mobility prediction, 2015.

[11] P Mazumdar, B K Patra, K S Babu et al., Hidden location prediction using check-in patterns in location-based social networks, *Knowledge & Information Systems*, vol.3, pp. 1-31, 2017.

[12] B Mao, J Cao, Z Wu, et al., Predicting driving direction with weighted markov model, *Advanced Data Mining and Applications*, pp. 407-418, 2012.

[13] Y Nagata, Population diversity measures based on variable-order markov models for the traveling salesman problem, *International Conference on Parallel Problem Solving from Nature. Springer International Publishing*, pp. 973-983, 2016.

[14] KVLMM: a trajectory prediction method based on a variable-order markov model with kernel smoothing, *IEEE Access [J]*, vol. 6, no. 1, pp. 25200-25208, 2018.

[15] J Yuan, X Liu, R Zhang, et al., Discovering semantic mobility pattern from check-in data, *International Conference on Web Information Systems Engineering. Springer, Cham*, pp. 464-479, 2014.

[16] Y J Kim, S B Cho, A HMM-based location prediction framework with location recognizer combining k-nearest neighbor and multiple decision trees, 2013.

[17] J Wu, Z M Cui, P P Zhao, et al.d Traffic vehicle behavior prediction using hidden markov models, *Artificial Intelligence and Computational Intelligence. Springer Berlin Heidelberg*, pp. 383-390, 2012.

[18] S J Qiao, T R Li, N Han et al., Adaptive trajectory prediction model for moving objects in large data environment, *Journal of Software*, vol. 26, no. 11, pp. 2869-2883, 2015.

[19] S Qiao, D Shen, X Wang, et al., A self-adaptive parameter selection trajectory prediction approach via hidden markov models, *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 284-296, 2015.