LETTER

Fast Time-Aware Sparse Trajectories Prediction with Tensor Factorization

Lei ZHANG†, Member, Qingfu FAN†, Guoxing ZHANG†, and Zhizheng LIANG†, Nonmembers

SUMMARY Existing trajectory prediction methods suffer from the "data sparsity" and neglect "time awareness", which leads to low accuracy. Aiming to the problem, we propose a fast time-aware sparse trajectories prediction with tensor factorization method (TSTP-TF). Firstly, we do trajectory synthesis based on trajectory entropy and put synthesized trajectories into the original trajectory space. It resolves the sparse problem of trajectory data and makes the new trajectory space more reliable. Then, we introduce multidimensional tensor modeling into Markov model to add the time dimension. Tensor factorization is adopted to infer the missing regions transition probabilities to further solve the problem of data sparsity. Due to the scale of the tensor, we design a divide and conquer tensor factorization model to reduce memory consumption and speed up decomposition. Experiments with real dataset show that TSTP-TF improves prediction accuracy generally by as much as 9% and 2% compared to the Baseline algorithm and ESTP-MF algorithm, respectively.

key words: trajectory prediction, data sparsity, tensor factorization

1. Introduction

We are increasingly benefiting from location based services with the widespread use of Global Positioning System (GPS). Trajectory prediction has become a hot topic [1] in recent years. Some researches based on probabilistic models, such as Markov models [2], pattern mining models [3] and spatio-temporal patterns [4], [5], have been proposed to predict the next place of individuals' mobility. As these methods estimate the users' future locations using a sequence of history locations, they suffer from the data sparsity problem: the original trajectory space contains many irregular trajectories and varied trajectory which make no historical trajectory match the query trajectory. Previous researches of Zheng [6]–[8] have shown that synthesized trajectories can alleviate the data sparsity problem. However, the prediction accuracy would be reduced as abnormal trajectories influence the reliability of synthesized trajectories. Aiming to this problem, we have proposed sparse trajectory prediction method based on entropy estimation (STP-EE) [9] and entropy-based sparse trajectories prediction enhanced by matrix factorization (ESTP-MF) [10]. Different from the above related works [5], [7], STP-EE uses trajectory entropy to evaluate trajectory’s regularity and chooses trajectories with lower entropy to reduce the number of abnormal trajectories. On the basis of STP-EE, ESTP-MF introduces matrix factorization into Markov models to solve the transition probability data sparsity problem. However, this method has a drawback. The time awareness is neglected due to it uses the single transition matrix in a day. The transition matrix in morning time is different from that in returning home time as traffic flow changes at every time slot, which affects the accuracy of the prediction.

In order to solve the above problem, we propose a fast time-aware sparse trajectories prediction with tensor factorization method (TSTP-TF). On the basis of ESTP-MF, TSTP-TF introduces tensor modeling and factorization to improve the sparse trajectory prediction. The main contributions of this paper are shown as follows:

1) In order to solve the problem of ignoring the time factor, we add the time dimension and construct the four-dimensional trajectory transition probability tensor.

2) To solve the problem of transition probability tensor data sparsity, we adopt tensor factorization to infer the missing regions transition probabilities.

3) It results in insufficient memory due to the scale of the tensor. We design a divide and conquer tensor factorization model to reduce memory consumption and speed up decomposition.

2. Sparse Trajectory Prediction

We present the architecture of TSTP-TF in Fig. 1, which consists of four layers: 1) trajectory serialization, 2) trajectory entropy estimation, 3) modeling trajectory transition tensor, 4) trajectories prediction enhanced by tensor factorization.

2.1 Trajectory Synthesis Based on Entropy Estimation

Firstly, we serialize the original GPS points as trajectories. An origin trajectory can be presented by \( n \) coordinate points with timestamp.

\[
tra = \{ (t_i, lon_i, lat_i) | t_i < t_{i+1} \}_{i=1}^{n}
\]  

(1)

where \( t_i, lon_i, lat_i \) denote the \( i \)th point’s time, longitude and latitude. The map is constructed as a two-dimensional grid graph which consists of \( G \) by SIGP [12]. All coordinate points are chronologically mapped to the grid graph so that a trajectory is represented as a sequence of grids according to the sequence of locations. Formally,
where $g_i$ is the grid graph of trajectory sequence at timeslot $t_i$. Due to there are some irregular trajectories in the original trajectory space, we implement the L-Z entropy estimation [9] to evaluate the regularity of the trajectory. For each trajectory $tra_i$, the entropy $e_i$ of $tra_i$ is computed and the trajectories are sorted by entropy value $\{ (tra_i, e_i) | e_i < e_{i+1} \}_{i=1}^{n}$. Then, the trajectories which have comparatively low entropy value are chosen. For every chosen trajectory, if there are cross-nodes with other trajectories, divide them into sub-trajectories by these cross-nodes. Then computing the sub-trajectories entropy by L-Z entropy estimation. The sub-trajectories are sorted by the sequence of nodes of the trajectory that is going to be synthesized. Keeping the sub-trajectories has lower entropy if there is overlapping among them. Finally, the remainder sub-trajectories with lower entropy are synthesized.

3. Fast Time-Aware Sparse Trajectories Prediction with Tensor Factorization

3.1 Time-Aware Four-Dimensional Transition Probability Tensor

As ESTP-MF uses the single transition matrix in a day to neglect the time awareness. The transition matrix in morning time is different from that in returning home time as traffic flow changes at every time slot. Weekdays and weekends also have different situations. It affects the accuracy of the prediction. So, we add the time dimension (including the hour dimension and the working day dimension) to the transition matrix, and construct the four-dimensional trajectory transition probability tensor. As shown in Fig. 2, we model the trajectory transition in each region using a tensor, $Y \in \mathbb{R}^{F \times T \times S \times W}$ with four dimensions denoting from regions $(F)$, to regions $(T)$, time slots $(S)$, Category $(W)$.

From region dimension $(F)$: This denotes regions $g = \{ g_1, g_2, \ldots, g_f, \ldots, g_N \}$. To region dimension $(T)$: This denotes regions $g = \{ g_1, g_2, \ldots, g_t, \ldots, g_T \}$.

Time slot dimension $(S)$: A day is divided into equal time slots $s = \{ s_1, s_2, \ldots, s_h, \ldots, s_S \}$.

Category dimension $(W)$: This denotes the weekday or weekend $w = \{ w_1, w_2 \}$.

An entry: An entry $Y(f, t, s, w)$ stores the transition probability $p_{fi}^{sw}$ which means that a user from region $g_f$ during timeslot $s_h$ and category $w_i$ moves to a destination region $g_t$ during timeslot $s_h + 1$. The transition probability is computed as Eq. (3):

$$p_{fi}^{sw} = \frac{\text{count}[g_f(s_h, w_i) \rightarrow g_t(s_{h+1}, w_i)]}{\sum \text{count}[g_f(s_h, w_i) \rightarrow g_s(s_{k}, w_j)]}$$

where $g_s$ is the tag of any location at $(s_{h+1}, w_i)$. As the transition probability tensor is very sparse, we adopt tensor factorization to infer the missing regions transition probabilities.

3.2 A Divide and Conquer Tensor Factorization Model (DC-TF)

Tensor factorization factorizes a large tensor into low-rank matrices to approximate its missing elements. As the scale of the tensor, it is too complicated for computation complexity and space to result in insufficient memory. So, we design a divide and conquer tensor factorization model to reduce memory consumption and speed up decomposition. DC-TF partitions the tensor data into sub-tensors, then conducts ALS on each sub-tensor independently as shown in Fig. 3. Tensor $Y$ is partitioned into two sub-tensors $Y_1$ and $Y_2$, each sub-tensor can be decomposed into sub-factor $M_{bi}^{(1)}$ independently. $Y = Y_{\overline{P}} | \overline{P} \in P$, where $P$ is the set of sub-tensor indexes. For each $Y_{\overline{P}}$, $Y_{\overline{P}} = A \times_1 M_{bi}^{(1)} \times_2 M_{bi}^{(2)} \times_N M_{bi}^{(N)}$. Where $A$ is the $N$-mode $F \times F \times \cdots \times F$ identity tensor. Partition each factor $M_{bi}^{(i)}$ into $P_i$ parts corresponding to the block boundaries along mode $i$, $M_{bi}^{(i)} = [M_{bi}^{(1)} M_{bi}^{(2)} \ldots M_{bi}^{(K_i)}]$. \(\overline{P} = [p_1, \ldots, p_i, \ldots, p_N] \in P\), then $Y_{\overline{P}} = A \times_1 M_{p_1}^{(1)} \times_2$
The current estimate of the sub-factor \( M_{(p)} \) can be revised using the \( T_{(p)}^{(i)} \) where \( T_{(p)}^{(i)} = \sum_{\mathcal{T} \in \{ \cdots , p_i , \cdots \}} M_{(p)}^{(i)}(A^{(i)} T (M_{(p)}^{(i)} M_{(p)}^{(i)})), S_{(p)}^{(i)} = \sum_{\mathcal{T} \in \{ \cdots , p_i , \cdots \}} Q^{(i)} \odot (M_{(p)}^{(i)} M_{(p)}^{(i)})) \). where \( \mathcal{T} = \{ i_1 , i_2 , \cdots , i_N \}, A_{(p)} = \otimes_{h=1}^{n} (M_{(h)}^{(i)} M_{(h)}^{(i)}), Q_{(p)} = \otimes_{h=1}^{n} (M_{(h)}^{(i)} M_{(h)}^{(i)}). \) Above, \( \otimes \) denotes the Hadamart product and \( \odot \) denotes the element-wise division operation. The input tensor is partitioned into several blocks in this way and each block can be decomposed with the available memory. So, it can handle large tensors. DC-TF is described as follows.

**Algorithm 1.** DC-TF algorithm

Input: tensor \( Y \), divide pattern \( P \), factorization rank \( F \)

Output: new tensor \( Y \)

1. for \( Y = \{ Y_{\mathcal{T}_{\mathcal{P}}} | \mathcal{P} \in P \} // \mathcal{P} \) is the set of sub-tensors index
2. decompose \( Y_{\mathcal{T}_{\mathcal{P}}} \) into \( M_{\mathcal{T}_{\mathcal{P}}}^{(1)}, M_{\mathcal{T}_{\mathcal{P}}}^{(2)}, \cdots, M_{\mathcal{T}_{\mathcal{P}}}^{(N)} // \mathcal{P} = \{ p_1, p_2, \cdots, p_N \} \in P \)
3. repeat
4. for each mode \( i = 1 \) to \( N \)
5. for each partition, \( p_i = 1 \) to \( P_i \)
6. update \( M_{(p)}^{(i)} \leftarrow T_{(p)}^{(i)} S_{(p)}^{(i)} \) using \( M_{(p)}^{(i)} \), for each block \( Y_{p_1 \cdots p_i \cdots p_N} \); more specifically
7. compute \( T_{(p)}^{(i)} = \sum_{\mathcal{T} \in \{ \cdots , p_i , \cdots \}} M_{(p)}^{(i)} (A^{(i)} T (M_{(p)}^{(i)} M_{(p)}^{(i)}))) \)
8. revise \( A_{p_1 \cdots p_i \cdots p_N} \) using \( M_{(p)}^{(i)} \) and \( M_{(p)}^{(i)} \)
9. compute \( S_{(p)}^{(i)} = \sum_{\mathcal{T} \in \{ \cdots , p_i , \cdots \}} Q^{(i)} \odot (M_{(p)}^{(i)} A_{(p)}^{(i)})) \) using the above
10. update \( M_{(p)}^{(i)} \) using the above
11. for each \( \mathcal{P} = \{ \cdots , p_i , \cdots \} \)
12. update \( A_{\mathcal{P}} \) and \( Q_{\mathcal{P}} \) using \( T_{(p)}^{(i)} \) and \( A_{(p)}^{(i)} \)
13. until stopping condition
14. Return \( Y \)

The lines 1–2 generated the sub-factors \( M_{(1)}^{(i)}, M_{(2)}^{(i)}, M_{(3)}^{(i)} \) of sub-tensor \( Y_1 \). The sub-factors \( M_{(i)}^{(i)} \) of each \( M_{(p)}^{(i)} \) are used for iteratively refining the sub-factors \( M_{(p)}^{(i)} \) of the input tensor \( Y \) in the lines 4–13.

### 3.3 Trajectories Prediction

Let probability vectors \( A_{(p)} \) represent distributions of visit probabilities of grids during timeslots \( s_h \) and category \( u_t \), \( A_{(s_h,u_t)} = \{ A_{(s_h,u_t)}^1, A_{(s_h,u_t)}^2, \cdots, A_{(s_h,u_t)}^n \} \). The visit probability denoted by \( A_{(s_h,u_t)}^i \) of a grid \( g_i \in G \) is a numerical estimate of the likelihood that users will visit grid \( g_i \) during \( s_h \in S \) and category \( u_t \). The value of visit probability is computed as Eq. (4):

\[
A_{(s_h,u_t)}^i = \frac{P(g_i)P((s_h,u_t)|g_i)}{P(g_i)P((s_h,u_t)|g_i) + \sum_{g_n \in G, g_i \neq g_n} P(g_n)P((s_h,u_t)|g_n)}
\]

where \( 0 \leq A_{(s_h,u_t)}^i \leq 1 \), \( P(g_i) \) is a visit probability of grid \( g_i \) in terms of two component probabilities, \( P((s_h,u_t)|g_i) \) is a conditional probability that \( g_i \) is visited occurred during \( (s_h,u_t) \cdot \sum_{g_i \in G} A_{(s_h,u_t)}^i = 1 \)

Given an initial probability distribution and the tensor of transition probabilities, the prediction destination of a target query trajectory is calculated using Eq. (5):

\[
P[(g_{pre}, (s_{h+1}, u_{t+1})) | (g_t, (s_h, u_t))] = \arg \max_s \left\{ \sum_{g_i \in G} A_{(s_h,u_t)}^i p_{st} \right\}
\]

### 4. Experiments and Analysis

We use a real-world large scale taxi trajectory dataset from the T-drive [11] project in experiments. It contains a total of 580,000 taxi trajectories in Beijing, 15 million GPS points from February 2, 2008 to February 8, 2008. We ran experiments with a weaker configuration, consisting of quad-core Intel(R) Core(TM) i5-2400 CPU @3.1GHz machines with 8 GB RAM.

#### 4.1 Evaluation on the Method

We compare Prediction Accuracy and the standard deviation of the prediction accuracy of the TSTP-TF with five approaches, Baseline algorithm coined from [2], SubSyn [6], STP-EE [9], STP-EM [12] and ESTP-MF [10] under different grid granularity \( n \) ranging from \( 10^8 \) to \( 10^3 \). We select 80% trajectories in the dataset as a training dataset and the remainder 20% trajectories as test dataset and divide every day into twelve equal time slots. The Prediction Accuracy (PA) is computed as the ratio between the number of correctly predicted trajectories and the total number of trajectories. As the trajectory is represented as a sequence of grids according to the sequence of locations \( g_i \), the accuracy is measured on grid values.
In this paper, we have proposed the TSTP-TF to import time-aware and tensor factorization to sparse trajectory prediction. Experiments have shown that this method has successfully addressed the data sparsity. Compared with the Baseline algorithm, SubSyn, STP-EE, STP-ME and ESTP-MF, TSTP-TF has higher prediction accuracy.

5. Conclusion

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities (2017XKQY078).

References

[1] L. Zhang, Z. Hu, and G. Yang, “Trajectory outlier detection based on multi-factors,” IEICE Tans. Inf. & Syst., vol.E-97D, no.8, pp.2170–2173, Aug. 2014.
[2] S. Gambs, M.-O. Killijian, and M.N. del Prado Cortez, “Next place prediction using mobility Markov chains,” Proc. 2012 ACM Conf. First Workshop on Measurement, Privacy, and Mobility (MPM), pp.1–6, New York, USA, April 2012.
[3] W. Groves, E. Nunes, and M. Gini, “A framework for predicting trajectories using global and local information,” Proc. 11th ACM Conf. Computing Frontiers, no.37, Cagliari, Italy, May 2014.
[4] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, “Trajectory pattern mining,” Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.330–339, San Jose, USA, Aug. 2007.
[5] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, “WhereNext: A location predictor on trajectory pattern mining,” Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.637–646, Paris, France, June 28–July 1, 2009.
[6] A.Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, “Destination prediction by sub-trajectory synthesis and privacy protection against such prediction,” Proc. 2013 IEEE International Conf. Data Engineering (ICDE), pp.254–265, Brisbane, Australia, April 2013.
[7] A.Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li, “Solving the data sparsity problem in destination prediction,” J. VLDB, vol.24, no.2, pp.219–243, 2015.
[8] Y. Zheng, “Trajectory data mining: An overview,” ACM Trans. Intelligent Systems and Technology, vol.6, no.3, Article No.29, pp.1–41, 2015.
[9] L. Zhang, L.J. Liu, and W. Li, “Sparse trajectory prediction based on entropy estimation,” IEICE Tans. Inf. & Syst., vol.E99-D, no.6, pp.1474–1481, June 2016.
[10] L. Zhang, Q. Fan, W. Li, Z. Liang, G. Zhang, and T. Luo, “Entropy-based sparse trajectories prediction enhanced by matrix,” IEICE Tans. Inf. & Syst., vol.E100-D, no.9, pp.2215–2218, Sept. 2017.
[11] T-drive trajectory data sample, M. Research, Aug. 2011, available from http://research.microsoft.com/apps/pubs/?id=152883.
[12] L. Zhang, L. Liu, Z. Xia, W. Li, and Q. Fan, “Sparse trajectory prediction based on multiple entropy measures,” J. Entropy, vol.18, no.9, 2016.
[13] J. Jeon, E.E. Papalexakis, U. Kang, and C. Faloutsos, “HaTen2: Billion-scale tensor decompositions,” Proc. 15th ICDE, pp.1047–1058, 2015.