IEICE TRANS. INF. & SYST., VOL.E103–D, NO.1 JANUARY 2020

LETTER

Mode Normalization Enhanced Recurrent Model for Multi-Modal Semantic Trajectory Prediction

Shaojie ZHU[†,††a)], ***Nonmember***, **Lei ZHANG**[†,††b)], ***Member***, **Bailong LIU**[†,††], **Shumin CUI**[†,††], **Changxing SHAO**[†,††], ***and*** **Yun LI**[†,††], ***Nonmembers***

SUMMARY Multi-modal semantic trajectory prediction has become a new challenge due to the rapid growth of multi-modal semantic trajectories with text message. Traditional RNN trajectory prediction methods have the following problems to process multi-modal semantic trajectory. The distribution of multi-modal trajectory samples shifts gradually with training. It leads to difficult convergency and long training time. Moreover, each modal feature shifts in different directions, which produces multiple distributions of dataset. To solve the above problems, MNERM (Mode Normalization Enhanced Recurrent Model) for multi-modal semantic trajectory is proposed. MNERM embeds multiple modal features together and combines the LSTM network to capture long-term dependency of trajectory. In addition, it designs Mode Normalization mechanism to normalize samples with multiple means and variances, and each distribution normalized falls into the action area of the activation function, so as to improve the prediction efficiency while improving greatly the training speed. Experiments on real dataset show that, compared with SERM, MNERM reduces the sensitivity of learning rate, improves the training speed by 9.120 times, increases HR@1 by 0.03, and reduces the ADE by 120 meters.

key words: *multi-modal, semantic trajectory, mode normalization*

1. Introduction

Trajectory prediction has become important in location-based applications. With the development of mobile Internet, social media has produced mass multi-modal semantic trajectory data. Unlike ordinary GPS trajectories, user preferences and text information describing user activities are added to reflect user intents and have great potential in improving trajectories prediction accuracies.

Traditional trajectory prediction methods, such as Hidden Markov Model [1] and Matrix Factorization [2], neglect the long-term dependency of the trajectory. Neural networks can effectively deal with long-term dependency. ST-RNN [3] models the temporal relationship combining the spatiotemporal law and recurrent neural network. LSTM (Long Short-Term Memory) network [4] is used to solve the long-term dependency in trajectory. Convolutional neural networks can also effectively extract the spatial regularity of trajectories [5]. The above methods only consider temporal and spatial features neglecting semantic features of trajectory.

There are few researches on multi-modal semantic trajectory prediction. SERM (Semantics-Enriched Recurrent Model) [6] combines multiple modal features (user, location, time, text) to predict trajectory. There is a drawback in SERM. With the training, the distribution of multi-modal trajectory samples shifts and falls gradually to the derivative saturation region of activation function. Therefore, the convergence time increases greatly as the parameters are always updated with a small gradient.

In order to solve the problem, we propose MNERM (Mode Normalization Enhanced Recurrent Model) for multi-modal semantic trajectory. Considering that the migration direction of each modal feature is not consistent, so the dataset has multiple distributions and using a single distribution to normalize is not effective. We introduce the MN (Mode Normalization) mechanism [7] to monitor the distribution of multi-modal trajectory sample in real-time by gating function, and standardize the trajectory samples by using the estimators of corresponding patterns. After normalized, the mean and variance of the sample fall into the action area of the activation function, and the corresponding derivative is far from the saturation area. Thus the model maintains a large gradient to update the parameters, which reduces the sensitivity of learning rate, ensures that the model can quickly converge and improves the effectiveness of prediction.

2. Mode Normalization Enhanced Recurrent Model

Given a set of grid-indexes $L = \{l_1, l_2, \ldots, l_{D_M}\}$ and a set of users $U = \{u_1, u_2, \ldots, u_{D_u}\}$. Where, D_M and D_u are the total number of meshes and users. A multi-modal semantic trajectory sequence of u_i is defined as $T(u_i) = \{r_1(u_i), \ldots, r_k(u_i), \ldots, r_K(u_i)\}$, $\forall 1 \le k < K$. $r_k(u_i) \in T(u_i)$ is a triple (t_k, l_k, c_k). Where, t_k is the timestamp, $l_k \in L$ is the grid-index and c_k is the text message.

As follows, MNERM consists of four parts:

(1) Joint embedding of multi-modal semantic trajectory features. A Multi-modal semantic trajectory includes grid-index, timestamp and text. For $r_k(u_i) = (t_k, l_k, c_k)$, we transform t_k, l_k and c_k into feature vectors e_{t_k}, e_{l_k}, e_{c_k} by embedding matrix E_t, E_l and E_c. $t_k \in R^{48}$ represents dividing a week into 48 periods, $c_k \in R^V$ represents a bag of keywords from a V-dimensional vocabulary, and $l_k \in R^{D_M}$.

Manuscript received July 5, 2019.
Manuscript revised September 7, 2019.
Manuscript publicized October 4, 2019.

[†]The authors are with School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, 221116, China.
[††]The authors are with Engineering Research Center of Mine Digitalization, Ministry of Education, Xuzhou 221116, China.
a) E-mail: 1208162499@qq.com
b) E-mail: zhanglei@cumt.edu.cn

DOI: 10.1587/transinf.2019EDL8130

The dimension of E_t, E_l and E_c is $D_t \times 48$, $D_l \times D_M$ and $D_c \times V$, where D_t, D_l, D_c are the dimension of the embeddings. By concatenating these embeddings, we obtain $e_k \in R^{D_e}$, $D_e = D_t + D_l + D_c$.

(2) LSTM capturing long-term dependency. For $T(u_i)$, the time step of LSTM is K. We take e_k as the input of k-th time step, then calculate the hidden state h_k as (1):

$$h_k \leftarrow f(W \cdot h_{k-1} + G \cdot e_k + b) \tag{1}$$

Where, $h_k \in R^{D_h}$, D_h denotes the number of hidden neurons and h_{k-1} is the previous hidden state. The involved parameters are: $D_h \times D_h$ matrix W, $D_h \times D_e$ matrix G and bias term $b \in R^{D_h}$.

(3) Mode normalization for trajectory samples.

(4) User representation and parameter learning. (3) and (4) will be explained in detail in subsequent chapters.

3. Mode Normalization for Trajectory Samples

The dimension of hidden state sequence $x = \{h_1, h_2, \ldots, h_K\}$ is $K \times D_h$, which is recorded as $\mathcal{X}$. $\{x_1, \ldots, x_n, \ldots, x_N\}$ is the batch samples sequence, N denotes the number of samples in each batch.

Each sample is firstly classified by a set of gating functions $\{g_1, \ldots, g_k, \ldots, g_Q\}$, where g_k: $\mathcal{X} \rightarrow [0, 1]$ and $\sum_1^Q g_k(x) = 1$. x_n is voted by its gate assignment as (2):

$$g_{n_k} \leftarrow [\sigma \circ \Psi(x_n)]_k \tag{2}$$

Where Ψ is an affine transformation: $\mathcal{X} \rightarrow R^Q$ and it is followed by a softmax activation σ: $R^Q \rightarrow [0, 1]^Q$.

We determine new component-wise statistics for each distribution: $N_k \leftarrow \sum_n g_{n_k}$, $\langle x \rangle_k \leftarrow \frac{1}{N_k} \sum_n g_{n_k} x_n$, $\left\langle x^2 \right\rangle_k \longleftarrow \frac{1}{N_k} \sum_n g_{n_k} x_n^2$. The running estimates are updated in each iteration with a memory parameter $\lambda \in (0, 1]$, as (3) and (4):

$$\overline{\langle x \rangle_k} \leftarrow \lambda \langle x \rangle_k + (1 - \lambda) \overline{\langle x \rangle_k} \tag{3}$$

$$\overline{\langle x^2 \rangle_k} \leftarrow \lambda \left\langle x^2 \right\rangle_k + (1 - \lambda) \overline{\langle x^2 \rangle_k} \tag{4}$$

The estimators for mean: $\mu_k \leftarrow \langle x \rangle_k$ and variance: $\sigma_k^2 \leftarrow \left\langle x^2 \right\rangle_k - \langle x \rangle_k^2$ are computed under weighing from the gating network. Each sample is then normalized as (5):

$$MN(x_n) \triangleq \alpha \left(\sum_{k=1}^{Q} g_k(x_n) \frac{x_n - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} \right) + \beta \tag{5}$$

Where α and β are learned parameters, $\mathcal{E}$ is a small const to avoid zero denominator. The test samples were normalized by using the running average component-wise estimators $\mu_k \longleftarrow \overline{\langle x \rangle_k}$ and $\sigma_k^2 \leftarrow \overline{\langle x^2 \rangle_k} - \overline{\langle x \rangle_k^2}$ of the training set. The sample $x_z = \{h_1, h_2, \ldots, h_K\}$ is transformed to $y_z = \left\{\overline{h_1}, \overline{h_2}, \ldots, \overline{h_K}\right\}$ after normalized, the dimension of y_z is also $\mathcal{X}$.

4. User Representation and Parameter Learning

In order to predict the final location, firstly, we transform $\overline{h_k} \in R^{D_h}$ into $o_k \in R^{D_M}$ by $o_k \leftarrow H \cdot \overline{h_k} + a$. Secondly, we use $e_{u_i} \leftarrow E_u \cdot u_i$ to embed user preference. Thirdly, we combine e_{u_i} and o_k to $\overline{o_k}$ by $\overline{o_k} \leftarrow e_{u_i} + o_k$. Where, H is a $D_M \times D_h$ transformation matrix, $a \in R^{D_M}$ is a bias term and E_u is a $D_M \times D_u$ embedding matrix. Finally, we input $\overline{o_k}$ into softmax activation σ to get y_{z_k} by $y_{z_k} \leftarrow \sigma(\overline{o_k})$.

To train the MNERM and infer the parameters, we use cross entropy as the loss function. Given a training set with Z samples, we define the objective function as (6):

$$J = -\sum_{z=1}^{Z} \sum_{k=1}^{K-1} l_{k+1} \log(y_{z_k}) + \frac{\delta}{2} \|\Theta\|^2 \tag{6}$$

Where $\Theta = \{E_t, E_l, E_c, E_u, W, G, H, b, a\}$ denotes all the parameters to be estimated, and δ is a pre-defined const for regularization. We use Stochastic Gradient Descent and Back Propagation Through Time to learn the parameter set Θ.

5. Experiments and Analysis

5.1 Dataset, Evaluation Metrics and Experiment Contents

Our experiments are based on multi-modal semantic trajectory dataset of New York City. It consists of 3863 multi-modal semantic trajectories of 235 users. The code is completed under Python 2.7, Keras 2.2.4 and Tensorflow 1.5.0. Hardware is shown as: 12-core processor, 32G memory, NVIDIA Tesla P100 graphics card. MNERM is compared with SERM [6] and BNERM (Batch Normalization Enhanced Recurrent Model). Where, BNERM uses BN [8] in the normalization phase.

We randomly select 80% trajectories as the training data, and use the remaining 20% for testing. We use these metrics to evaluate the performance. (1) HR@k (Hitting Ratio @k) Examine whether the ground-truth location appears in the top-k result list. (2) ADE (Average Distance Error) Calculate the average distance error of ground-truth location and the top-5 result list. (3) CT (Convergence Time) It includes the number of epochs and OAET (One-step Average Epoch Time). We set the number of distributions $Q = 2$ for MNERM by several experiments. We set $D_l = D_t = D_c = D_h = 50$ and $N = 100$ for the three methods.

We design experiments as follows: (1) We set the LR of SERM model to 0.01, 0.001 and 0.0005. Then, we compare HR@k, ADE and CT of SERM under corresponding LR. We also compare the performance of different evaluation metrics of the three methods under $LR = 0.0005$. (2) Under $LR = 0.0005$, we compare the CT of the three methods when their HR@k and ADE reach global optimum. (3) We compare the optimal performance of HR@k and ADE for SERM, BNERM and MNERM.

5.2 Experimental Analysis

In Fig. 1, with the increase of LR, SERM converges faster, but the effectiveness of its HR@k and ADE fluctuates more

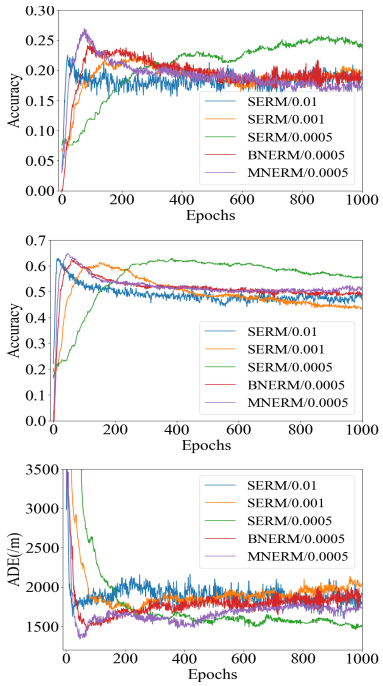

Fig. 1 The convergence curve of HR@1, HR@20 and ADE.

Table 1 The optimal performance of HR@k, ADE(m) for SERM, BNERM and MNERM under corresponding LR.

Method/*LR*	HR@1	HR@5	HR@10	HR@20	ADE(m)
SERM/0.01	0.2251	0.4264	0.5109	0.5964	1624
SERM/0.001	0.2229	0.4255	0.5230	0.6145	1616
SERM/0.0005	0.2571	0.4504	0.5489	0.6204	1457
BNERM/0.0005	0.2394	0.4528	0.5255	0.6259	1438
MNERM/0.0005	0.2694	0.4747	0.5633	0.6478	1337

obvious and decreases gradually. It proves that SERM is highly sensitive to LR. In Table 1, BNERM converges quickly with $LR = 0.0005$, but the accuracy of HR@1 and HR@10 are decreased by 0.0177 and 0.0234 than that of SERM. It indicates that although BNERM reduces the sensitivity of LR, the performance of some evaluation metrics is reduced. Also, the convergence speed of MNERM under $LR = 0.0005$ is only slightly slower than that of SERM under $LR = 0.01$, and the performance of MNERM/0.0005 is the best as is shown in Table 1.

In Table 1, compared with SERM/0.01 and SERM/0.001, SERM/0.0005 has the best performance of SERM. Also, BNERM/0.0005 and MNERM/0.0005 are the optimal performance of the corresponding models. So, compared with SERM, the HR@1, HR@5, HR@10 and HR@20 of MNERM increase by 0.0123, 0.0243, 0.0144 and 0.0274 respectively, and the ADE decreases by 120m.

We compare the CT of optimal performance for SERM, BNERM and MNERM. Table 2 shows that the OAET of BNERM and MNERM increases by 7s and 12s due to their increasing computing, but the training speed of MNERM

Table 2 The CT(s) of optimal performance and the OAET(s) for SERM, BNERM and MNERM under $LR = 0.0005$.

Method	HR@1	HR@5	HR@10	HR@20	ADE	OAET
SERM	45209	19080	18338	20299	51834	53
BNERM	5640	5400	4560	3600	4140	60
MNERM	4875	3770	3965	2925	3055	65

is 8.274, 4.061, 3.624, 7.939 and 15.967 times faster than that of SERM for all metrics due to less epochs. In order to measure the overall training speed, we compare the longest CT of all metrics of SERM and MNERM, and conclude that MNERM increases the training speed by 9.633 times as a whole.

6. Conclusion

We have proposed the MNERM for multi-modal semantic trajectory prediction. By allocating the trajectory samples to multiple distributions and normalizing them, MNERM can reduce the sensitivity of *LR* and increase the effectiveness of prediction at the same time fast prediction.

Acknowledgments

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

References

[1] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "GMove: Group-level mobility modeling using geo-tagged social media," Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.1305–1314, ACM, 2016.

[2] N. Duong-Trung, N. Schilling, and L. Schmidt-Thieme, "Near real-time geolocation prediction in twitter streams via matrix factorization based regression," Proc. 25th ACM International on Conference on Information and Knowledge Management, pp.1973–1976, ACM, 2016.

[3] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," 30th AAAI Conference on Artificial Intelligence, 2016.

[4] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," 2018 24th International Conference on Pattern Recognition (ICPR), pp.1941–1946, IEEE, 2018.

[5] J. Lv, Q. Li, Q. Sun, and X. Wang, "T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction," 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), pp.82–89, IEEE, 2018.

[6] D. Yao, C. Zhang, J. Huang, and J. Bi, "SERM: A recurrent model for next location prediction in semantic trajectories," Proc. 2017 ACM on Conference on Information and Knowledge Management, pp.2411–2414, ACM, 2017.

[7] L. Deecke, I. Murray, and H. Bilen, "Mode normalization," arXiv preprint arXiv:1810.05466, 2018.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.