Deep Learning of Pair-wise Interactions of Moving Objects Course Project for EECS6400

Soroush Sheikh Gargar
Project Supervisor: Manos Papagelis
Course Supervisor: Mokhtar Aboelaze

***Abstract* – Spatio-temporal data mining (STDM) is becoming growingly important in the big data era with the increasing availability and importance of the large spatio-temporal datasets such as maps, virtual globes, remote-sensing images, and GPS trajectories. Mining valuable knowledge from spatio-temporal data is critically important to many real-world applications including human mobility understanding, smart transportation, urban planning, public safety, health care, and environmental management.**
In the current project, we attempt to find a method for STDM using state of the art Deep Learning. We adapted one of the most reliable approaches in interaction learning and tried to use it to learn spatio-temporal data such as trajectories.

***Index Terms*—Multi-agent, Interactions, Spatio-temporal, Trajectory, Deep trajectory learning, Interaction Graph, Graph Learning, Trajectory Representation Learning, Interaction Learning, Evolving Graphs.**

I. Introduction

WITH the fast development of the various positioning systems such as Global Positioning System (GPS), mobile devices and remote sensing, spatio-temporal data has become increasingly available nowadays. STDM has broad applications in various domains including environment and climate (e.g., wind prediction and precipitation forecasting), public safety (e.g., crime prediction), intelligent transportation (e.g., traffic flow prediction), human mobility (e.g., human trajectory pattern mining), etc. Classical methods of data usually perform poorly in the case of spatio-temporal data. They are many reasons for such behavior. First, spatio-temporal data are usually embedded in continuous space, whereas classical datasets such as transactions and graphs are often discrete. Second, patterns of ST data usually present both spatial and temporal properties, which is more complex, and the data correlations are hard to capture by traditional methods. Finally, one of the common assumptions in traditional statistical-based data mining methods is that data samples are independently generated. When it comes to the analysis of spatio-temporal data, however, the assumption about the independence of samples usually does not hold because ST data tends to be highly self correlated.

One useful way to abstract the ST data is through the lens of interactions. As a motivating example, let us take the movement of cars in the street. It is clear that the dynamics of a single car are influenced by the other cars, and observing these dynamics as a human, we are able to reason about the different types of interactions that might arise, e.g., one car moves along with another car, or moving in front of it. It might be feasible, though tedious, to manually annotate certain interactions given a task of interest. It is more promising to learn the underlying interactions, perhaps shared across many tasks, in an unsupervised fashion.

In this work, we are using this view to face the problem of trajectory similarity, which is a fundamental problem in STDM. We use state-of-the-art Neural networks (in particular Graph neural networks) to build a representation of the trajectories in the graph domain where in that, trajectories which has the most interactions together (interaction could be proximity) are closer. The main Contributions of this work are as follow:

- Building a representation of trajectories in a graph domain where the trajectories which have most interactions form an edge in between.
- Use the representation to predict the future time steps of the trajectory which are unknown for the system
- Make a synthetic dataset, and evaluate the performance of the method with simple cases of trajectories

The rest of the report is as follow: in section II, we review some of the noteworthy works in the literature regarding the problem of trajectory similarity analysis. After that, we give a brief introduction of Graph Neural Networks in section III. in section IV we formally explain the problem statement and in section V we propose our method of representation learning for trajectory similarity learning. Section VI belong to the experiments and results. Finally, in section VII, we state some of the possible future routes for the research and conclude the report.

II. Related Works

The Problem of detecting similarity between trajectories of spatio-temporal data has been in the literature for a while now, and many came up with various methods to face it. We can categorize the literature into two sections:

A. Analytical Methods

these methods use classical non-deep-learning approaches such as dynamic time wrapping (DTW) [1], longest common subsequence (LCSS) [2], edit distance with real penalty (ERP) [3], and edit distance on real sequences (EDR) [4]. These existing methods usually try to form a pairwise matching the sample points of two trajectories and identify the best alignment using dynamic programming. More specifically, these pairwise point-matching methods implicitly partition the space into cells according to a threshold ϵ and match two points if they fall into the same cell. Dynamic programming is

then used to find an alignment that minimizes a matching cost. The pairwise point-matching methods for computing trajectory similarity often suffer in three scenarios: the first is when the sampling rates of trajectories are non-uniform. Sampling rates often vary across devices due to different device settings, battery constraints, and communication failures. Even for the same device, the sampling rates may vary. For example, a taxi driver may alter the default device sampling rate to reduce power consumption. [5] This may result in sampling points alternating between sparse and dense episodes. This poses challenges to the existing methodsif two trajectories represent the same underlying route, but are generated at different sampling rates, it is difficult for these methods to identify them as similar trajectories. The second scenario where it is challenging to align the sample points of similar trajectories is when the sampling rate is low. For example, the sampling rates for trajectories generated from online check-ins (e.g., Foursquare, Facebook), geo-tagged tweets, geo-tagged photo albums, and call detail records are low and inherently non-uniform [6]. Third, the performance of these methods may be degraded when the sample points are noisy. Such noise may occur in GPS points when moving in urban canyons. Finally, they need hand-crafted threshold settings for their point-wise interaction settings. This behavior causes a lack of generalization and also need for an expert in each case.

B. Deep Learning Based Methods

These methods benefits from inherent properties such as *Automatic feature representation learning* Significantly different from traditional machine learning methods that require hand-crafted features, deep learning models can automatically learn hierarchical feature representations from the raw ST data. In STDM, the spatial proximity and the long-term temporal correlations of the data are usually complex and hard to be captured. With the multi-layer convolution operation in CNN and the recurrent structure of RNN, such spatial proximity and temporal correlations in ST data can be automatically and effectively learned from the raw data directly.

III. Background: Graph Neural Networks

It is necessary for our work to give a brief introduction to the Graph Neural Networks (GNNs). Graph Neural Network is a type of Neural Network which directly operates on the Graph structure [7], given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $v \in \mathcal{V}$ and edges $e = (v, v') \in \mathcal{E}$. A single node-to-node message passing define as follow:

$$v \to e : h^l_{(i,j)} = f^l_e([h^i_i, h^l_j]) \tag{1}$$

$$e \to v : h^{l+1}_j = f^l_v([\sum_{i \in \mathcal{N}_j} h^l_{(i,j)}]) \tag{2}$$

h^l_j is the embedding of node v_i in layer l, $h^l_{(i,j)}$ is an embedding of the edge $e_{(i,j)}$. $\mathcal{N}_j$ denotes the set of indices in the neighborhood of v_j and $[.,.]$ denotes the concatenation of vectors. The functions f_v and f_e are node and edge specific neural networks (e.g. small MLPs) respectively. Equations (1)-(2) allow for the composition of the models that map from different edge to node representation and vice-versa via multiple rounds of message passing. Figures 1 and 2 illustrate how a GNN works.

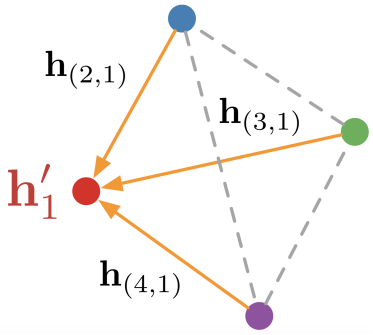

Fig. 1. Message passing operation where $h_{(2,1)}, h_{(3,1)}, h_{(4,1)}$ calculated by Eq. 1 and h'_1 calculated by Eq. 2

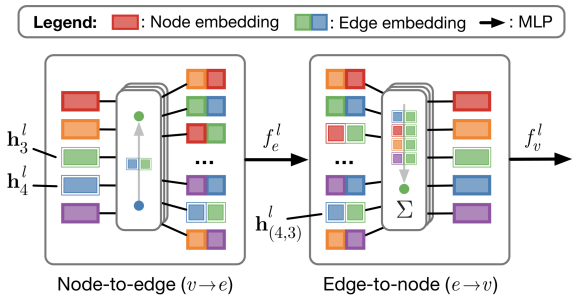

Fig. 2. Node-to-edge ($v \to e$) and edge-to-node ($e \to v$) operations for moving between node and edge representations in a GNN. $v \to e$ represents concatenation of node embeddings connected by an edge, whereas $e \to v$ denotes the aggregation of edge embeddings from all incoming edges. In our notation in Eqs. (1)(2), every such operation is followed by a small neural network (e.g. a 2-layer MLP), here denoted by a black arrow. For clarity, we highlight which node embeddings are combined to form a specific edge embedding ($v \to e$) and which edge embeddings are aggregated to a specific node embedding ($e \to v$).

IV. Problem Statement

One of the most important spatio-temporal data representation is trajectories. Trajectories denote the paths traced by bodies moving in space over time. (e.g., the moving route of a bike trip or taxi trip). Trajectory data are usually collected by the sensors deployed on the moving objects that can periodically transmit the location of the object over time, such as GPS on a taxi.

The inputs to our problem are these trajectories. More formally, our input consists of trajectories of N objects. We denote by x^t_i the feature vector of object v_i at time t, e.g. location and velocity. We denote by $x^t = \{x^t_1, ..., x^t_N\}$ the set of features of all N objects at time t, and we denote by $x_i = (x^1_i, ..., x^T_i)$ the trajectory of the object i, where T is the total number of time steps. Lastly, we mark the whole trajectories by $x = (x^1, ..., x^T)$.

The output is an interaction graph. Each trajectory is embedded as a node in the interaction graph and the weight of an edge between two nodes determines by how long and how much close they were together, the more time they spend close to each other the higher the weight of the edge in between. If

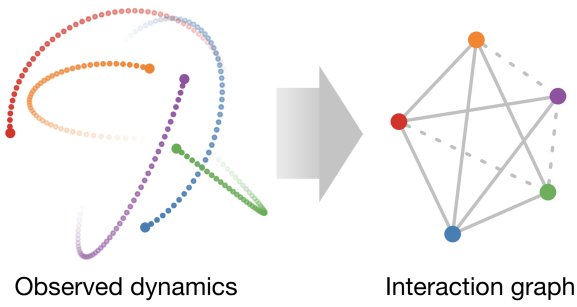

Fig. 3. given the trajectories as input, we aim to learn a graph representation where each node represent a trajectory and there is an edge between two nodes if the corresponding trajectories to those nodes have been in each others proximity for a significant amount of time

they have not spent any time close to each other, the weight of the edge between them should be zero. At the end one can make a binary graph using a threshold function. In this scenario, the weights become the probability of being an edge in the final binary graph. Figure 3 shows a scheme of the problem.

In total, we aim to learn a representation of trajectories in the graph domain and we can categorize this work in trajectory representation learning.

V. Proposed Method

A. Formulation and core Idea

After reviewing deep learning-based methods in trajectory similarity problem, we chose Neural Relational Inference (NRI) [8] as the target for our trajectory similarity learning. The model consists of two parts trained jointly. An encoder that predicts the interactions given the trajectories, and a decoder that learns the dynamical model given the interaction graph. The encoder returns a factorized distribution of z_{ij} (the probability of edge type between two objects of v_i and v_j) which is essentially the representation that we want to learn. The decoder models the probability of the features for the next time step of the trajectory given the current known trajectory and the representation we learned with the encoder using a GNN. The model is formalized as a variational autoencoder (VAE) [9] that maximize the ELBO:

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})] \tag{3}$$

At a high level, the goal of the encoder is to infer pairwise interaction types z_{ij} given observed trajectories $x = (x^1, ..., x^T)$. More formally, The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ returns a factorized distribution of $\mathbf{z}_{ij}$, where $\mathbf{z}_{ij}$ is a discrete categorical variable representing the edge types, given the trajectories $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_T)$ where K is number of objects. More formally, the encoder is modelled by $q_\phi(\mathbf{z}|\mathbf{x}) = \mathrm{softmax}(f_{enc,\phi}(x)_{ij,1:K})$ where $f_{enc,\phi}(x)$ is a GNN acting on a fully connected graph.

$$h_j^1 = f_{emb}(\mathbf{x}_j) \tag{4}$$

$$v \to e : h_{(i,j)}^1 = f_e^1([h_i^1, h_j^1]) \tag{5}$$

$$e \to v : h_j^2 = f_v^1(\sum\nolimits_{i \neq j} h_{(i,j)}^1) \tag{6}$$

$$v \to e : h_{(i,j)}^2 = f_e^2([h_i^2, h_j^2]) \tag{7}$$

Finally, $q_\phi(\mathbf{z}_{ij}|\mathbf{x}) = \mathrm{softmax}(h_{(i,j)}^2)$, where ϕ summarize all the trainable parameters of the neural network in the equations (4)-(7)

The functions $f(...)$ are neural networks that map between the respective representations. In our experiments we used either fully-connected networks (MLPs) or 1D convolutional networks (CNNs) with attentive pooling.

The decoder

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z}) \tag{8}$$

is trying to predict future given the latent graph structure and past time steps of the trajectory $\mathbf{z}$. If the dynamic is Markovian $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z}) = p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, \mathbf{z})$ the model uses a separate GNN for reconstructing the physical interactions for each edge type as follow:

$$v \to e : \tilde{h}_{(i,j)}^t = \sum_k z_{ij,k} \tilde{f}_e^k([x_i^t, x_j^t]) \tag{9}$$

$$e \to v : \mu_j^{t+1} = x_j^t + \tilde{f}_v(\sum\nolimits_{i \neq j} \tilde{h}_{(i,j)}^t) \tag{10}$$

$$p(x_j^{t+1}|x^t, z) = \mathcal{N}(\mu_j^{t+1}, \sigma^2 \mathbf{I}) \tag{11}$$

$z_{ij,k}$ is the k-th element of the vector $\mathbf{z}_{ij}$ and σ^2 is a fixed variance Figure 4 is showing the overall architecture of the model.

B. Training

For training the GNNs given the trajectories $\mathbf{x}$ first, one should run the encoder and compute $q_\phi(\mathbf{z}_{ij}|\mathbf{x})$, then sample $\mathbf{z}_{ij}$ from the concrete reparametrizable approximation of $q_\phi(\mathbf{z}_{ij}|\mathbf{x})$. Then run the decoder to compute μ. The ELBO objective in Equation (3) consist of two terms : the reconstruction error $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ and the KL divergence $\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})]$. The former is estimated by:

$$-\sum_j \sum_{t=2}^{T} \frac{\left\|\mathbf{x}_j^t - \mu_j^t\right\|^2}{2\sigma^2} + \mathrm{const} \tag{12}$$

And KL term when we have a uniform prior is sum of entropies:

$$\sum_{i \neq j} H(q_\phi(\mathbf{z}_{ij}|\mathbf{x})) + \mathrm{const} \tag{13}$$

with this reparametrization one can compute the gradients by back-propagation and optimize.

VI. Experiments and Results

A. Experimental Setup and Datasets.

For the experiments we first used a synthetic dataset in which each sample consist of three objects moving a 10x10 box. Two of them moving in close proximity of 0.01 unit, and one of them moves with a distance (the starting position of this object has chosen randomly over the box). The number of the samples in the data set is 50,000, and each sample consist of the trajectory in 49 time steps. At each time step, the feature

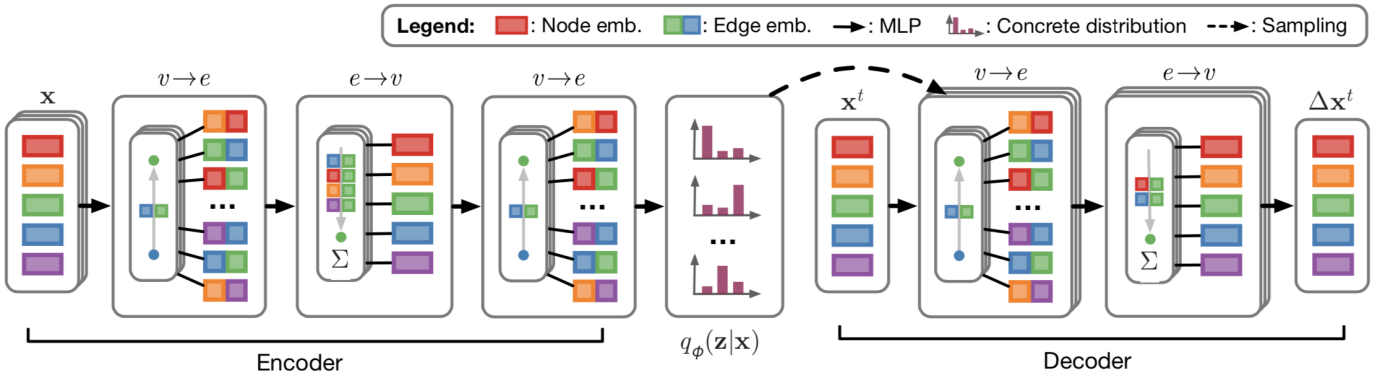

Fig. 4. The encoder is trying to predict the probability distribution of the latent graph-based model $q_\phi(\mathbf{z}|\mathbf{x})$ given the trajectories, with a GNN. The decoder trying the reconstruct the interactions in the real domain given both latent model and current state of the objects by running several GNNs in parallel for each edge type.

vector of the object has two values reporting its location in the box and also its velocity. Since our objective is to learn a representation based on the proximity of the moving objects, we expect to get an interaction graph where in which there is an edge between the two close trajectories and no other edge. We also performed the method over two dynamic interaction dataset of physical phenomenon Spring5 and Charged5. The former belongs to 5 objects which some of them are connected with springs, and the latter consist of 5 moving charged particles having interaction together. The NRI model originally developed to predict the future time-steps of the simple physical phenomenon, and we took these two datasets from the original work in the [8].

We used MLP and CNN in the encoder with following architectures:

- **MLP**: 2-layer MLP with hidden and output dimension of 256, with batch normalization, dropout, and ELU activations.
- **CNN**: The CNN encoder uses another block which performs 1D convolutions with attention

We also used the same MLP architecture for the decoder.

All experiments were run using the Adam optimizer [10] with a learning rate of 0.0005, decayed by a factor of 0.5 every 200 epochs. The chosen batch size for the experiment was 128

B. Results and Analysis

The metrics we used to evaluate the performances are Edge Accuracy and Mean Square Error. Edge accuracy is the portion of the edges the model got correctly:

$$\text{Accuracy} = \frac{\text{number of edges the model predicted correctly}}{\text{number of all the edges}}$$

Mean Square Error is the average of the error in the prediction of the next 10 time steps of the trajectory.

Table I and II summarize the results

The results illustrate that the model works very good on simple physical phenomena. However, it can not be generalized to the general trajectory similarity problem. Our guess about this behavior is the nature of this physical phenomena.

TABLE I
Results on Synthetic Dataset

Encoder Type	Edge Accuracy	MSE
MLP	43.6%	4.1e-4
CNN	40.4%	4.1e-4

TABLE II
Results on Physical Phenomena Datasets

Dataset	Encoder Type	Edge Accuracy	MSE
Spring5	MLP	99%	3.29e-6
Charged5	CNN	82%	3.21e-3

Both Spring5 and charged5 consist of some repetitive actions which we can not see in the synthetic dataset. One other guess that can be investigated is that our encoder can only infer Markovian trajectories. Both experimented physical dataset benefit from the Markovian underlying ground-truth where we can cot get in general trajectory problem. In the end, one can say that although this method works very well for some special cases, it can not be generalized to the overall trajectory problem.

Despite the poor performance in learning a representation based on proximity, the model performs surprisingly well on predicting future time steps. And from this, we can say that the predictive ability of the model does not have a strong correlation with the learned representation from the encoder and it is mainly based on the input from the known past time steps of the trajectory.

VII. Conclusion and Future Works

With the increase of location based data and raise of the IoT and smart cities, as well as many other location based evolving big data. There is a crucial need for mining the spatio-temporal data in order to increase efficiency. Currently, many big companies like Uber and Lyft can benefit from spatio-temporal data inference for their business. Moreover, they are not the only applications. Many other sectors of the society like environment preservation and health care can use

these Inferences as well.

The state-of-the-art deep learning has affected many other branches of computer science and data mining; trajectory representation learning is no exception. There have been many attempts to make a framework based on deep learning to analyze ST data, and this work was one of them.

However, as we mentioned in the previous section the methodology that we chose does not generalize well to the problem of conventional trajectories, and it is only useful for specific cases like repetitive phenomena.
The other problem with this method rather than lack of generalization is that it is not easily scalable. It needs a significant amount of resources, for training. With our available resources we could perform the method on the at most 5 particles for a consistent result. So performing it on the rich datasets such as pedestrians in the city or vehicles traffic with our current infrastructures is not possible.
For the future works, since the general trajectory problem is not Markovian, we think about using RNN-based encoder to capture the sequential nature of the trajectories. And, in case of not getting promising result from that, we will change the baseline to other deep learning based trajectory similarity analysis methods.

REFERENCES

[1] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings 14th International Conference on Data Engineering*. IEEE, 1998, pp. 201–208.

[2] M. Vlachos, D. Gunopoulos, and G. Kollios, "Discovering similar multidimensional trajectories," in *icde*. IEEE, 2002, p. 0673.

[3] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 792–803.

[4] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 491–502.

[5] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 1144–1155.

[6] S. Ranu, P. Deepak, A. D. Telang, P. Deshpande, and S. Raghavan, "Indexing and matching trajectories under inconsistent sampling rates," in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 999–1010.

[7] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[8] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," *arXiv preprint arXiv:1802.04687*, 2018.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.