Asynchronous Message Passing on Dynamic Graphs

Keyue Jiang

1. Introduction

While most graph machine learning methods have been targeted at static graphs, many important real-world graphs are dynamic – they change with time. For example, infectious diseases spread along with the changing graph of interpersonal contacts. Graphs of social networks and e-commerce also change as new members join, or new data becomes available.

In spite of the clear importance of dynamic graphs, research and techniques for such data are not as advanced as that of static graphs. Existing works have largely focused on translating static graph concepts to dynamic graphs with only limited success, and it is generally agreed that more fundamental advances are needed [4].

In the proposal, we argue that the paradigm of most existing Graph Neural Networks (GNNs), so-called synchronous message passing process [10], is not ideal for modeling dynamic graphs. Thus, we then propose to use Asynchronous Message Passing (AMP) [2] models instead to overcome the challenges in dynamic graph machine learning. We believe the AMP models can better model the time-varying nature and solve the information redundancy problem of dynamic graphs.

2. Background and Challenges

There are mainly two types of dynamic graphs:

- **Discrete-time dynamic graphs** (DTDG) sample a sequence of static graph snapshots with a fixed time interval T, and the graph at timestamp t would be formalized as $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, where $\mathcal{V}_t$ and $\mathcal{E}_t$ are respectively the sets of nodes and edges.
- **Continuous-time dynamic graphs** (CTDG) represents the whole process with a list of events $I_L = \{S_1, S_2, \dots, S_i, \dots S_L\}$. Each interaction event is defined as $S_i = (u_i, v_i, t_i, \mathbf{f}_i)$, where $i \in \{1, 2, \cdots, L\}$. t_i is the timestamp, u_i and v_i are the nodes participated in the event and $\mathbf{f}_i$ is the context feature. It is possible to construct a graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ by treating the events **before** t as edges, and the event participators as nodes [18].

It can be shown that CTDG records the exact temporal information when an event happens, which grants a more fine-grained representation of the dynamic graphs. The data of CTDG can be reshaped into the format of DTDG, but the reshaping operation would always lose the context information.

The classification of dynamic graphs gave birth to two main approaches for modeling. First, a wide range of models to process DTDG takes the form of combining GNN and RNN [1, 5–8, 12, 15]. E.g., [7, 15] simply fed the output of GNN into RNN; EvolveGCN [12] used the RNN to regulate the GNN parameters. On the other hand, the research on CTDG is still in its infant stage with only a few papers [3, 14, 16, 17]. Their ideas are to encode the continuous time into the node embedding and optimize the model at each event.

However, the aforementioned methods commonly suffer from two problems:

Model Expressiveness: The models based on GNN follow the synchronous message passing framework [10], and these GNN architectures commonly experience the problem of oversmoothing, underreaching and oversquashing [11]. The situation of underreaching and oversquashing is even worse in the dynamic graphs[1].

[1] well.. not theoretically proved but I encountered them in my previous research.

Even though there are bunches of solutions proposed recently (e.g. the ideas in distributed computing [9,10]), they do not translate well in dynamic graphs.

Information Redundancy: The existing methods all treat the graph dynamics to be incremental and ignore the inactivation of nodes/edges. i.e., when a model is predicting node-level features, it will aggregate all the messages sent to it. But in dynamic graphs, the long-term (temporal) and long-range (spatial) messages are not always helpful in the current prediction, and the existing models all fail to identify this redundant information. This nature would lead to expensive computational costs and terrible prediction performance.

So we would ask: When a node is communicating with others in dynamic graphs, is it possible to control the long-range and long-term message flow to avoid information redundancy, while still preserving high model expressiveness?

3. Methodology

The potential solution lies in the utilization of Asynchronous Message Passing (AMP) [2] in the field of dynamic graph machine learning. AMP is a concept adapted from distributed computing [13], and has already been used in the design of GNN architectures [2]. In this proposal, we suggest that AMP models can also work as a new paradigm for dynamic graphs, and could potentially solve the expressiveness limitation and information redundancy in modeling graph dynamics.

We consider the scenario of CTDG, where the whole process is represented as a series of events $\{S_i\}_{i\in 1,\dots L}$. The dynamics of the new method are very similar to the AMP paper [2], which can be described as follows.

1. Each node will be initialized with a state h. Different from the model in AMP paper [2], the node could potentially emit a message when receiving a message, or **participating in an event**. The first message m_0 will be generated spontaneously as the event S_0 happens at t_0.

2. At timestamp t_i, upon receiving a message m or **participating in an event** S_i, the node can react to it and change its internal state.

3. Then the node can decide whether or not to emit another new message m' to its neighborhoods. The message will also contain the information about the event (e.g. encode an event embedding by an MLP). If the action is triggered by receiving message m, the probability of emitting a new message can be controlled by a gated unit. So it is possible to cut off the long-term or long-range message flow explicitly.

4. Finally, when all the events finish, the hidden state of nodes can be used for downstream tasks, e.g. link prediction, node classification etc.

The design intends to identify the redundant information contained in those long-term and long-range messages, and prevent them from propagating through the graph. I feel that this paradigm could achieve ideal performance in many tasks on dynamic graphs. However, there are a few problems that I haven't successfully formalized:

- Whether or not to consider the communication delay, i.e. to process the message immediately when it's emitted or set a delay frame before processing. This might help us to control the temporal message fading-away as well.

- The message could contain the information about source node and the time of emission. So it might also be possible to introduce the distance (between the current node and the source node) and the time to control the probability of message emission.

References

[1] Chen, J., Xu, X., Wu, Y., and Zheng, H. GC-LSTM: graph convolution embedded LSTM for dynamic link prediction. *CoRR abs/1812.04206* (2018).

[2] Faber, L., and Wattenhofer, R. Asynchronous neural networks for learning in graphs. *CoRR abs/2205.12245* (2022).

[3] Goyal, P., Kamra, N., He, X., and Liu, Y. Dyngem: Deep embedding method for dynamic graphs. *CoRR abs/1805.11273* (2018).

[4] Holme, P., and Saramäki, J. *Temporal network theory*, vol. 2. Springer, 2019.

[5] Jin, W., Qu, M., Jin, X., and Ren, X. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530* (2019).

[6] Li, J., Han, Z., Cheng, H., Su, J., Wang, P., Zhang, J., and Pan, L. Predicting path failure in time-evolving graphs. In *KDD* (2019), ACM, pp. 1279–1289.

[7] Manessi, F., Rozza, A., and Manzo, M. Dynamic graph convolutional networks. *Pattern Recognit. 97* (2020).

[8] Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *ICML* (2016), vol. 48 of *JMLR Workshop and Conference Proceedings*, JMLR.org, pp. 2014–2023.

[9] Papp, P. A., Martinkus, K., Faber, L., and Wattenhofer, R. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. In *NeurIPS* (2021), pp. 21997–22009.

[10] Papp, P. A., and Wattenhofer, R. An introduction to graph neural networks from a distributed computing perspective. In *ICDCIT* (2022), vol. 13145 of *Lecture Notes in Computer Science*, Springer, pp. 26–44.

[11] Papp, P. A., and Wattenhofer, R. A theoretical comparison of graph neural network extensions. *CoRR abs/2201.12884* (2022).

[12] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., and Leiserson, C. E. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI* (2020), AAAI Press, pp. 5363–5370.

[13] Peleg, D. *Distributed Computing: A Locality-Sensitive Approach.* Society for Industrial and Applied Mathematics, USA, 2000.

[14] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. M. Temporal graph networks for deep learning on dynamic graphs. *CoRR abs/2006.10637* (2020).

[15] Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. In *ICONIP (1)* (2018), vol. 11301 of *Lecture Notes in Computer Science*, Springer, pp. 362–373.

[16] Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. Dyrep: Learning representations over dynamic graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019), OpenReview.net.

[17] Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. In *ICLR* (2020), OpenReview.net.

[18] Zhang, M., Wu, S., Yu, X., and Wang, L. Dynamic graph neural networks for sequential recommendation. *CoRR abs/2104.07368* (2021).