Marked Neural Spatio-Temporal Point Process Involving a Dynamic Graph Neural Network

Alice Moallemy-Oureh[*], **Silvia Beddar-Wiesing**, **Rüdiger Nather**, **Josephine Thomas**
University of Kassel, 34121 Kassel, Germany
{amoallemy, s.beddar-wiesing, r.nather, jthomas}@uni-kassel.de

Abstract

Spatio-Temporal Point Processes (STPPs) have recently become increasingly interesting for learning dynamic graph data since many scientific fields, ranging from mathematics, biology, social sciences, and physics to computer science, are naturally related and dynamic. While training Recurrent Neural Networks and solving PDEs for representing temporal data is expensive, TPPs were a good alternative. The drawback is that constructing an appropriate TPP for modeling temporal data requires the assumption of a particular temporal behavior of the data. To overcome this problem, Neural TPPs have been developed that enable learning of the parameters of the TPP. However, the research is relatively young for modeling dynamic graphs, and only a few TPPs have been proposed to handle edge-dynamic graphs. To allow for learning on a fully dynamic graph, we propose the first Marked Neural Spatio-Temporal Point Process (MNSTPP) that leverages a Dynamic Graph Neural Network to learn Spatio-TPPs to model and predict any event in a graph stream. In addition, our model can be updated efficiently by considering single events for local retraining.

1 Introduction

Graph Neural Networks (GNNs) have become baseline models for learning on structured data in the past years. Many models have been developed for static graph data that can use structural information in the learning process of various problems [15]. However, many applications produce more complex data and problems, necessitating the incorporation of temporal information, which static graphs cannot capture [19]. Therefore, learning on dynamic graphs has been growing for several years. Models have been proposed that can handle node attribute changes [13], additions of nodes or edges (growing graphs) [16, 17], or edge-structure dynamics and node attribute changes [18]. More examples are provided in [9, 15]. This, however, demonstrates that most models are specialized for processing certain dynamic graphs.

Approaches to modeling dynamics in graphs can be divided into three main categories: The usage of Recurrent Neural Networks (RNNs), Partial Differential Equations (PDEs), or Temporal Point Processes (TPPs). However, the drawback of RNNs is that explicit temporal information is neglected, and only the sequence of changes in the graph is represented. Furthermore, solving PDEs is costly and non-trivial, making TPPs more appropriate for representing graph stream data. However, they have the disadvantage of making assumptions about the temporal patterns that are not necessarily met. Therefore, Neural TPPs have been proposed [4, 10], which can efficiently learn the temporal evolution of complex processes without requiring prior knowledge. Though only a few models for graphs have been developed exploiting them, they are also limited to certain graph types.

In this work in progress, we present the first Marked Neural Spatio-Temporal Point Process (MNSTPP) utilizing a Dynamic GNN to learn on graph streams with arbitrary structural and attribute dynamics.

[*] All contributions are listed in detail in the appendix App. A.4.

Temporal Graph Learning Workshop @ NeurIPS 2023, New Orleans.

The approach is based on the model DyREP [17], which applies a Neural TPP on edge-growing graphs. We extend the Neural TPPs of DyREP to Marked Neural Spatio-Temporal Point Processes, which support processing any structural changes in graphs. The representation of continuous node and edge attributes in the form of a marked TPP with real-valued marks is innovative compared to the usage of non-marked TPPs or categorical marks as carried out in most of the literature so far [2, 20] and [6, 10], or marks only for non-graph data [3, 7, 8, 21]. By creating the Neural TPPs based on a Dynamic GNN, we can predict all kinds of structural events, event times, and attribute changes over time. Furthermore, the model is updateable for every new event by local retraining, which enables a fast and efficient update and training of the model.

2 Preliminaries

We first introduce preliminary ideas that constitute essential modules in the model. **Graph Streams.** A **graph stream** $G = (\mathcal{G}_0 := (\mathcal{V}_0, \mathcal{E}_0, \alpha_0, \beta_0), \mathcal{O})$ is a dynamic graph in continuous-time representation. Given by a static start graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0, \alpha_0, \beta_0)$ with nodes $\mathcal{V}_0$, edges $\mathcal{E}_0$ and attribute mappings $\alpha_0 : \mathcal{V}_0 \to \mathcal{A}$, $\beta_0 : \mathcal{E}_0 \to \mathcal{B}$ with arbitrary attribute sets $\mathcal{A}, \mathcal{B}$, and events $o_{k,t} \in \mathcal{O}$ of the following form:

Table 1: Events in a continuous-time representation of a graph.

Event Type	Addition	Deletion	Attribute Change
Node Event	$o_{0,t} := (v, \alpha_t^v, t)_{\text{add}}$	$o_{1,t} := (v, \alpha_t^v, t)_{\text{del}}$	$o_{4,t} := (v, \alpha_t^v, t)_{\text{atr}}$
Edge Event	$o_{2,t} := (e, \beta_t^e, t)_{\text{add}}$	$o_{3,t} := (e, \beta_t^e, t)_{\text{del}}$	$o_{5,t} := (e, \beta_t^e, t)_{\text{atr}}$

Here, $v \in \mathcal{V}_{\leq t} := \bigcup_{\hat{t}\,:\,\hat{t}\leq t} \mathcal{V}_{\hat{t}}$ are nodes and $e \in \mathcal{E}_{\leq t} := \bigcup_{\hat{t}\,:\,\hat{t}\leq t} \mathcal{E}_{\hat{t}}$ edges appearing in G up to timestamp t. Further, $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ is a set of timestamps and we assume here that only one event occurs at each timestamp.

Activities of Nodes and Edges. Dealing with structure dynamics, we use a so-called activity function to capture which nodes/edges exist at any given time. To learn the graph's deletion behavior, the activity of deleted nodes/edges is 0, and for the addition behavior, we use 1. Formally, let $o_t(x)$ be an event at time t that affects the item $x \in \mathcal{V}_t \cup \mathcal{E}_t$, and $\xi : (\mathcal{V} \cup \mathcal{E}) \times \mathcal{T} \to \{0, 1\}$ be the **activity function** defined as

$$\xi_t^x = \begin{cases} 1, & \text{if } o_t(x) \in \{o_0, o_2, o_4, o_5\} \ \wedge \ x \in \mathcal{V}_t \cup \mathcal{E}_t \quad \text{addition and attribute change events} \\ 0, & \text{if } o_t(x) \in \{o_1, o_3\} \ \wedge \ x \in \mathcal{V}_t \cup \mathcal{E}_t \quad \text{deletion events (inactivity)} \\ 0, & \text{if } x \notin \mathcal{V}_t \cup \mathcal{E}_t. \end{cases} \tag{1}$$

The proposed model uses the activity values to determine realistic observations, e.g., just existing nodes can be deleted.

Attribute Embedding. The attribute embedding is determined by a Recurrent Neural Network (RNN) encoding the dynamic behavior of the node/edge attribute considering the historical attribute information in the embedding vector $\boldsymbol{u}_{\bar{t}}(\gamma_t^x)$ and the current attribute vector γ_t^x given by a preprocessing module, as, e.g., a CNN for images, text2vec for text [12] etc. Then, the update scheme of an attribute γ_t^x of item x at timestamp $t \in \mathcal{T}$ with learnable parameters $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ is given by

$$\boldsymbol{u}_t^x = \boldsymbol{w}_0 \cdot (\boldsymbol{u}_{\bar{t}}^x, \gamma_t^x)^\top + \boldsymbol{w}_1. \tag{2}$$

Marked Temporal Point Processes. Given an event stream, it is non-trivial to tell when a new observation will occur and what it may look like. Given that the observation is dependent on the historical point patterns, the underlying process can be modeled by a **(Marked) Temporal Point Process (TPP)** [11].

Definition 2.1 (Marked Temporal Point Process) *Let $\boldsymbol{H} = \{(t_i, \boldsymbol{m}_i)\}_{i=0}^n$ be a sequence of $n > 0$ observations $\boldsymbol{m}_i \in \mathbb{R}^d$ at timestamps t_i, and $\boldsymbol{H}_t = \{(t_i, \boldsymbol{m}_i) \mid t_i < t, \ (t_i, \boldsymbol{m}_i) \in \boldsymbol{H}\}$ entail the history of events before timestamp t. Then, the **conditional intensity function***

$$\lambda(t, \boldsymbol{m} \mid \boldsymbol{H}_t) = \lim_{\Delta t \downarrow 0, \Delta \boldsymbol{m} \downarrow 0} \frac{\mathbb{P}\left(t_i \in [t, t + \Delta t], \boldsymbol{m}_i \in B(\boldsymbol{m}, \Delta \boldsymbol{m}) \mid \boldsymbol{H}_t\right)}{|B(\boldsymbol{m}, \Delta \boldsymbol{m})| \Delta t} \tag{3}$$

characterizes the MTPP completely [3], where $B(\boldsymbol{m}, \Delta\boldsymbol{m}) \subset \mathbb{R}^d$ is an open ball with center $\boldsymbol{m}$ and radius $\Delta\boldsymbol{m}$ for continuous events and $B(\boldsymbol{m}, \Delta\boldsymbol{m}) = \boldsymbol{m}$ for discrete events. The function is required to be non-negative.

Remark 2.2 (Spatio-Temporal Point Process) *If the observations $\boldsymbol{m}$ determine a location, the Spatio-Temporal Point Process describes the evolution of data in space and time [5].*

Neural Temporal Point Processes. The intensity functions of the different TPPs have to be chosen carefully based on assumptions about the appearance of temporal patterns in the data. Since these assumptions are not necessarily met in real-world data, partially and fully trainable Neural TPPs [8] have been proposed [4, 10] to learn the more complex temporal pattern in the data. In the proposed work, we utilize a *partially trainable Neural TPP*, so the definition of the fully-trainable Neural TPP is skipped here but can be found in [8].

Definition 2.3 (Neural Temporal Point Process) *Partially trainable Neural TPPs [8] replace intensity functions by Deep Neural Networks (DNNs) to encode the temporal information given by the event history $\boldsymbol{H}_{t_i}$. Then, the resulting feature vector $\boldsymbol{h}_{t_i}$ is forwarded to a selected temporal decay function ϕ which together determine the intensity*

$$\lambda(t_i, \boldsymbol{m}_i \mid \boldsymbol{H}_{t_i}) = \phi(\Delta t_i, \Delta\boldsymbol{m}_i \mid \boldsymbol{h}_{t_i}) = \phi(\Delta t_i, \Delta\boldsymbol{m}_i \mid \mathrm{DNN}(\boldsymbol{H}_{t_i})). \tag{4}$$

Here, $\Delta t_i = t_i - \bar{t}_i$ and $\Delta\boldsymbol{m}_i = \|\boldsymbol{m}_i - \bar{\boldsymbol{m}}_i\|$ are the deviations of the timestamps and marks from the events at t_i and the last event before it.

Attention. The idea of a Graph Attention Neural Network (GAT) is to incorporate scaling factors for neighboring nodes in the neighborhood aggregation process in GNNs. In this work, we leverage the attention mechanism from the model GATv2 [1] for the **neighborhood propagation** to scale the neighboring nodes $z \in \mathcal{N}(x)$ of nodes or edges x at time t with

$$q_t^{\{x,z\}} = \mathrm{softmax}(\boldsymbol{w_0}^\top \mathrm{LeakyReLU}(\boldsymbol{W}_1 \cdot [\boldsymbol{h}_x^t \| \boldsymbol{h}_z^t])), \tag{5}$$

where $\boldsymbol{h}_x^t$ are the hidden representations of the items. For the **self-propagation**, we utilize the mechanism as temporal attention for one item x as $p_t^x = \mathrm{softmax}(\boldsymbol{w_0}^\top \mathrm{LeakyReLU}(\boldsymbol{W}_1 \cdot [\boldsymbol{h}_x^t \| \boldsymbol{h}_x^{\bar{t}}]))$.

3 Model

The proposed model first utilizes a Dynamic GNN for incorporating historical local and attribute information of nodes and edges to obtain expressive embeddings. Afterward, Neural Temporal Point Processes determine the distribution of the different structural and attribute events over time, relying on the node and edge embeddings. The training method and possible predictions are explained in appendix App. A.3.

3.1 Dynamic Graph Neural Network

The hidden representations of nodes and edges are calculated similarly to a GAT [1] with additional consideration of the embedding evolution in the self-propagation module, the attribute embedding (cf. Eq. (2)) and the temporal delay of events on the node/edge in the exogenous drive.

Node Embedding. To incorporate the entire information given at a node, the embedding update for a node includes the self-propagation, neighborhood propagation, exogenous drive, and attribute embedding modules. The **self-propagation** comprises the evolution of the node embedding $\boldsymbol{Z}_t^v$ for a node v at time $\bar{t}$ over time weighted with temporal attention as described in (5). In the **local embedding propagation** scheme, the hidden representations and attribute embeddings of the neighboring nodes $\mathcal{N}(v) = \{w \mid \{v, w\} \in \mathcal{E}\}$ are linearly combined scaled by the attention factors of (5) by $\boldsymbol{h}_{loc}(v, t) = \sum_{w \in \mathcal{N}(v)} q_t^{\{v,w\}} \cdot \boldsymbol{Z}_t^w$. The **exogenous drive** includes the time difference between the timestamps of the current event and the one beforehand affecting a node at time t and $\bar{t}$, respectively. The last component includes the **attribute embedding** of the node as defined in (2). Summarized, the modules result in the node embedding determined by

$$\boldsymbol{Z}^v(t) = \sigma\left(\underbrace{\boldsymbol{M}_1 \boldsymbol{h}_{loc}(v,t)}_{\text{loc. embd. prop.}} + \underbrace{\boldsymbol{M}_2 \boldsymbol{Z}_{\bar{t}}^v \cdot p_t^v}_{\text{self-prop.}} + \underbrace{\boldsymbol{m}_3 \cdot (t - \bar{t})}_{\text{exogenous drive}} + \underbrace{\boldsymbol{M}_4 \cdot \boldsymbol{u}(v,t))}_{\text{attribute prop.}}\right).$$

The edge embedding is obtained analogously. The only substantial difference is the choice of the local embedding propagation scheme. You can find more information on that and the initial embeddings of both in the appendix App. A.1.

3.2 Neural Temporal Point Processes

The temporal evolution of the events is captured with the aid of different TPPs, dependent on the event type. For structural changes in the graph, we introduce spatio-temporal intensity functions following the approach of a partially trainable Neural TPP as described in (4) using the node and edge embeddings introduced earlier in Sec. 3.1. Considering real-valued attribute changes, we utilize Marked TPPs from (3) and the attribute embeddings from (2).

Structural Changes. Spatio-Temporal Point Processes describe events' distribution, including a spatial component over time. Space is determined as a node or an edge in the graph sequence. The TPP is then defined via the spatio-temporal intensity function $\lambda_k(t, x \mid \boldsymbol{h}_t) = \phi_k(g_k(t, x, \boldsymbol{h}_t))$, which consists of concatenating a selected temporal function ϕ_k and a learnable scoring function g_k based on the Dynamic GNN. As temporal activation function, the non-negative decay function $\phi_k(x) = \psi_k \log(1 + \exp(x/\psi_k))$ with timescale parameter $\psi_k > 0$ is used, similar to the choice in [17]. The scoring function g_k is designed to suit different cases according to the event type k and realizes the Dynamic GNN for valid conditions based on the activity function of (1).

For example, the intensity of a **node addition** at time t is 0 if the node already exists, i.e., $\xi_t^v = 1$. Otherwise, the event is scored by forwarding the current node embedding $\boldsymbol{Z}_t^v$ through a linear layer $g_0^t(v) = \boldsymbol{W}_0^\top \boldsymbol{Z}_t^v$.

The intensities for **node deletion and edge addition/deletion** are formalized analogously and can be found in the appendix App. A.2.

Attribute Changes. Attributes can only change if and only if the item x whose attributes are in consideration exists, i.e., $\xi_t^x = 1$. If this is the case, a Marked Neural TPP is necessary whose scoring function considers the embedding changes in $\boldsymbol{Z}_t^x$ in addition to the change of the attribute embedding $\boldsymbol{u}_t^x$ to capture the temporal and the structural information between $\bar{t}$ and t at item x. For the **node attribute change** event, e.g., the score is given by

$$g_4^t(v) = \boldsymbol{W}_4^\top \begin{pmatrix} \boldsymbol{u}_t^v \\ \boldsymbol{u}_{\bar{t}}^v \end{pmatrix} + \bar{\boldsymbol{W}}_4^\top \begin{pmatrix} \boldsymbol{Z}_t^v \\ \boldsymbol{Z}_{\bar{t}}^v \end{pmatrix}.$$

The **edge attribute change** is formalized analogous and can be found in the appendix App. A.2.

4 Conclusion and Future Work

The proposed model enables learning on graph streams, including arbitrary types of dynamics. The utilized Dynamic GNN processes the historical structural and temporal information and provides hidden representations of the nodes and edges. The subsequent different TPPs model the temporal behavior of the graph stream based on the hidden graph representation, and new occurring events can be efficiently integrated into the model.

However, our model is a work in progress and thus provides opportunities for further development. The model's reliability, explainability, and generalization are briefly discussed in the following.

Reliability. The proposed work still needs to be evaluated. Hence, extensive experiments must be conducted to prove the applicability and reliability of the model.

Explainability. GNNs are called explainable if the model explains the predicted result or reasoning that can be inferred based on the model architecture. In the future, the goal is to make our model more explainable to simplify the application of the model for real-world problems.

Definite Deletions vs. Inactivity. Furthermore, the activity function from Section 2 will be extended to consider definite deletions, activities, and inactivities of nodes/edges. Thereby, nodes and edges lose the possibility to occur again in the graph.

More complex graph types. The structural graph properties of the dynamic graphs in this paper are yet elementary, so the MNSTPP can be extended to more complex dynamical graph structures in the future, considering, e.g., the graph types described in [14].

References

[1] S. Brody, U. Alon, and E. Yahav. How Attentive are Graph Attention Networks? *arXiv preprint arXiv:2105.14491*, 2021.

[2] Chang, Xiaofu and Liu, Xuqin and Wen, Jianfeng and Li, Shuang and Fang, Yanming and Song, Le and Qi, Yuan. Continuous-Time Tynamic Graph Learning via Neural Interaction Processes. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 145–154, 2020.

[3] Chen, Ricky TQ and Amos, Brandon and Nickel, Maximilian. Neural Spatio-Temporal Point Processes. *arXiv preprint arXiv:2011.04583*, 2020.

[4] Du, Nan and Dai, Hanjun and Trivedi, Rakshit and Upadhyay, Utkarsh and Gomez-Rodriguez, Manuel and Song, Le. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1555–1564, 2016.

[5] Finkenstadt, B. and Held, L. and Isham, V. *Statistical Methods for Spatio-Temporal Systems*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2006.

[6] Gracious, Tony and Gupta, Arman and Dukkipati, Ambedkar. Neural Temporal Point Process for Forecasting Higher Order and Directional Interactions. *arXiv preprint arXiv:2301.12210*, 2023.

[7] H. Hachiya and S. Hong. Multistream-Based Marked Point Process With Decomposed Cumulative Hazard Functions. *Neural Computation*, 35(4):699–726, 2023.

[8] S. Hong and H. Hachiya. Multi-Stream Based Marked Point Process. In *Asian Conference on Machine Learning*, pages 1269–1284. PMLR, 2021.

[9] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[10] H. Mei and J. M. Eisner. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. *Advances in neural information processing systems*, 30, 2017.

[11] J. G. Rasmussen. Lecture Notes: Temporal Point Processes and the Conditional Intensity Function. *CoRR*, abs/1806.00221, 2018.

[12] Selivanov, Dmitriy and Bickel, Manuel and Wang, Qing. Package 'text2vec', 2020.

[13] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In L. Cheng, A. C. Leung, and S. Ozawa, editors, *Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*, volume 11301 of *Lecture Notes in Computer Science*, pages 362–373. Springer, 2018.

[14] J. M. Thomas, S. Beddar-Wiesing, A. Moallemy-Oureh, and R. Nather. Graph type expressivity and transformations. *CoRR*, abs/2109.10708, 2021.

[15] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter. Graph neural networks designed for different graph types: A survey. *CoRR*, abs/2204.03080, 2022.

[16] R. Trivedi, H. Dai, Y. Wang, and L. Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3462–3471. PMLR, 2017.

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

[18] D. Wang, Z. Zhang, Y. Ma, T. Zhao, T. Jiang, N. V. Chawla, and M. Jiang. Learning attribute-structure co-evolutions in dynamic graphs. *CoRR*, abs/2007.13004, 2020.

[19] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

[20] Xia, Wenwen and Li, Yuchen and Li, Shenghong. Graph Neural Point Process for Temporal Interaction Prediction. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4867–4879, 2022.

[21] Zhou, Zihao and Yang, Xingyi and Rossi, Ryan and Zhao, Handong and Yu, Rose. Neural Point Process for Learning Spatiotemporal Event Dynamics. In *Learning for Dynamics and Control Conference*, pages 777–789. PMLR, 2022.

A Appendix

A.1 Edge Embedding

Edge Embedding. The edge embedding is obtained analogously. The only substantial difference is the choice of the local embedding propagation scheme, which includes the embeddings of the incident nodes, their attributes, and the embedding of the edge attribute, namely

$$\boldsymbol{h}_{loc}(u,v,t) = \begin{bmatrix} \boldsymbol{Z}_{\bar{t}}^{u} & \boldsymbol{Z}_{\bar{t}}^{v} \\ \boldsymbol{u}(u,t) & \boldsymbol{u}(v,t) \end{bmatrix} \begin{bmatrix} p_{\bar{t}}^{u} \\ p_{\bar{t}}^{v} \end{bmatrix} \underbrace{\boldsymbol{Z}_{t}^{u} q_{t}^{u} + \boldsymbol{Z}_{t}^{v} q_{t}^{v}}_{\text{incident nodes cum.}}.$$

As a result we get the node embedding with

$$\boldsymbol{Z}^{u,v}(t) = \sigma\left(\underbrace{\boldsymbol{M}_7 \boldsymbol{h}_{loc}(u,v,t)}_{\text{loc. embd. prop.}} + \underbrace{\boldsymbol{M}_8 \boldsymbol{Z}_{\bar{t}}^{u,v} \cdot p_{t}^{\{u,v\}}}_{\text{self-prop.}} + \underbrace{\boldsymbol{m}_9 \cdot (t-\bar{t})}_{\text{exogenous drive}} + \underbrace{\boldsymbol{M}_{10} \cdot \boldsymbol{u}(\{u,v\},t))}_{\text{attribute prop.}} \right).$$

Initial Embedding. In case that the activity function of a node or edge x is undefined up to a timestamp t, the initial embedding only includes the attribute propagation and the local embedding propagation at time t. If a node has been added without neighbors, the local embedding propagation is empty and so the initial embedding only consists of the attribute propagation.

A.2 Remaining Intensity Functions

In Sec. 3.2, we introduced the intensity functions used to encode the different TPPs used in the model. Next to the node addition and the node attribute changes, further intensities are defined as follows:

- **Node Deletion:** The deletion of a node is defined analogously to the addition just that it is important to switch the cases, i.e., a node can just be deleted at timestamp t if and only if it existed before at timestamp $\bar{t}$. In this case the event is scored by $g_1^t(v) = \boldsymbol{W}_1^\top \boldsymbol{Z}_t^v$. Otherwise, the intensity is 0.
- **Edge Addition:** Analogously to the nodes additions, an edge e can only be added when it did not exist the timestamp before and, additionally, both nodes $u, v \in e$ exist. This time we score concerning both nodes and edges, $g_2^t(u,v) = \boldsymbol{W}_2^\top (\boldsymbol{Z}_t^u, \boldsymbol{Z}_t^v, \boldsymbol{Z}_t^e)^\top$. Otherwise, the intensity is 0.
- **Edge Deletion:** A deletion of an edge $e = \{u, v\}$ can only happen when the edge existed at time $\bar{t}$, or at least one of the incident nodes has been deleted. Thus, when the edge does not exist at time $\bar{t}$, it cannot be deleted, i.e., $\lambda_3^e(t) = 1 \iff \xi_t^{(u,v)} = 1 \ \wedge (\xi_t^u = 0 \vee \xi_t^v = 0)$. The impossible case is an intensity of 0. Otherwise, the scoring is $g_3^t(e) = \boldsymbol{W}_3^\top (\boldsymbol{Z}_t^u, \boldsymbol{Z}_t^v, \boldsymbol{Z}_t^e)^\top$.
- **Edge Attribute Change:** Here, the edge and the incident nodes must exist at time $\bar{t}$. Therefore, the scoring function considers a score for the occurrence of an event between the incident nodes and the edge attribute change analogously to the node attribute change, namely

$$g_5^t(e) = \boldsymbol{W}_5^\top \begin{pmatrix} \boldsymbol{Z}_t^u \\ \boldsymbol{Z}_t^v \\ \boldsymbol{Z}_t^e \end{pmatrix} + \bar{\boldsymbol{W}}_5^\top \begin{pmatrix} \boldsymbol{u}_t^e \\ \boldsymbol{u}_{\bar{t}}^e \end{pmatrix} + \bar{\bar{\boldsymbol{W}}}_5^\top \begin{pmatrix} \boldsymbol{Z}_t^e \\ \boldsymbol{Z}_{\bar{t}}^e \end{pmatrix}.$$

A.3 Training and Prediction

Depending on the application and the specific data set, there are different suitable learning techniques, and only the necessary intensity functions have to be included in the learning. Due to the sparse representation of the graph as a stream, the training and the update procedures are reasonably efficient.

Training. The training is performed similarly to the procedure described in [17]. Here, we have to extend the loss function to include the processing of continuous marks in the intensities. The learning is then conducted by

minimizing the negative log-likelihood

$$\mathcal{L} := -\sum_{o \in \mathcal{O}} \log\left(\Lambda(o)\right) + \underbrace{\int_0^T \left(\Theta_1(\tau) + \Theta_2(\tau)\right)\, d\tau}_{\text{survival probability}},$$

$$\text{where } \Lambda(o) = \begin{cases} \lambda_k^x(t), & \text{if } o \text{ is discrete event,} \\ \lambda_k^x(\boldsymbol{m}, t), & \text{if } o \text{ is continuous event} \end{cases}$$

$$\text{and } \Theta_1(\tau) = \sum_{\bar{o} \in \bar{\mathcal{O}}} \sum_{k=0}^{3} \lambda_k^x(\tau) \quad \text{intensities of discrete events}$$

$$\text{and } \Theta_2(\tau) = \sum_{\bar{o} \in \bar{\mathcal{O}}} \int_{\mathbb{R}^d} \lambda_k^x(\boldsymbol{m}, \tau) \quad \text{intensities of continuous events}$$

It represents the total probability for an event type and the attribute value the events come with. Here, $\bar{\mathcal{O}}(t)$ denotes a finite set of sampled events that did not happen until time t. For batch-wise training, the observation set $\mathcal{O}$ only includes the batch events.

Model Update. The model can be updated at an event by applying the mini-batch stochastic gradient descent on the negative log-likelihood, only considering the event and the affected neighboring nodes or edges. This local retraining approach minimizes the update effort as only a small subset of the parameters from the intensity functions have to be processed.

Prediction. For predicting a structural change at an item x in the event stream, the conditional density function

$$f_k^x(t) = \lambda_k^x(t) \cdot \exp\left(\int_{[\bar{t},t]} \lambda_k^x(\tau) \mathrm{d}\tau\right) \tag{6}$$

considering an appropriate intensity function λ_k provides the occurrence probability of an event of type k in the time interval $[\bar{t}, t]$. While using active or inactive nodes or edges does not cause any problems, adding new nodes that never existed in the graph event history Eq. (6) seems inappropriate, which we want to investigate further. To predict the time t_k^x when an event of type k occurs, the event prediction function $f_k^x(t)$ is integrated over the future by $\int_{[\bar{t},\infty]} t \cdot f_k^x(t)$. Predicting an attribute change of an item x by interpreting marks analogous to time, we use the conditional density

$$\boldsymbol{a}_x^\star = \int_{\mathbb{R}^d} \boldsymbol{a} \cdot \lambda_k^x(\boldsymbol{a}, t) \cdot \exp\left(\int_{[\bar{\boldsymbol{a}},\boldsymbol{a}]} \lambda_k^x(\boldsymbol{\alpha}, t) \mathrm{d}\boldsymbol{\alpha}\right) \mathrm{d}\boldsymbol{a}.$$

A.4 Contributions

- Conceptualization: SBW, AMO
- Methodology: SBW, AMO, RN
- Resources: AMO
- Writing (Original Draft): SBW, AMO
- Writing (Review & Editing): SBW, AMO, RN
- Supervision: RN, JT
- Project administration: SBW, AMO
- Funding acquisition: JT

Alice Moallemy-Oureh is mainly responsible for incorporating attribute dynamics, while Silvia Beddar-Wiesing is mainly responsible for modeling the structure dynamics.

A.5 Acknowledgement

The GAIN project is funded by the Ministry of Education and Research Germany (BMBF), under the funding code 01IS20047A, according to the 'Policy for funding female junior researchers in Artificial Intelligence'.
Further, the authors would like to thank Prof. Dr. Felix Lindner for the fruitful discussions and helpful feedback on our work.