Geometric Hawkes Processes with Graph Convolutional Recurrent Neural Networks for Modeling Temporal Events

Jin Shang, and Mingxuan Sun, *Member, IEEE*

***Abstract*—Hawkes processes are popular for modeling correlated temporal sequences that exhibit mutual-excitation properties. Existing approaches such as feature-enriched processes or variations of Multivariate Hawkes processes either fail to describe the exact mutual influence between sequences or become computational inhibitive in most real-world applications involving large dimensions. Incorporating additional geometric structure in the form of graphs into Hawkes processes is an effective and efficient way for improving model prediction accuracy. In this paper, we propose the Geometric Hawkes Process (GHP) model to better correlate individual processes, by integrating Hawkes processes and a graph convolutional recurrent neural network. The deep network structure is computational efficient since it requires constant parameters that are independent of the graph size. The experiment results on real-world data show that our framework outperforms recent state-of-art methods.**

***Index Terms*—Hawkes process, graph convolutional networks, temporal events.**

INTRODUCTION

Hawkes processes, which are capable of modeling temporal events that exhibit self-exciting properties, have been widely applied in various applications such as supporting decision making in smart health [1], inferring Granger causality [2], and predicting recurrent user behaviors [3], [4], [5]. Generally, Hawkes processes are useful for modeling a collection of correlated event sequences such as earthquakes at N locations or the diffusion of M infectious diseases among a group of N people. For example, in analyzing on-line user behaviors such as visiting websites, recent approaches such as [6] treat the recurrent events of each user-item pair as an one-dimensional Hawkes process, and assume the parameters of all processes have a low-rank structure. However, methods that typically treat each process independently would fail to achieve good performance when there are insufficient observations for each process.

Multivariate Hawkes processes [7] are suitable for modeling multiple correlated sequences, where the occurrence of an event in one sequence may influence the occurrence of new events in another. For example, in social event analysis, the events of an individual user can be modeled as an one-dimensional Hawkes process and events in a network can be modeled as a Multivariate Hawkes process [8], [9], [10], which captures the correlations of both endogenous and exogenous event intensities. Extensive studies [11], [12], [13], [2] have focused on estimating the excitation matrix of multivariate processes for different inference tasks. However, those approaches are either unable to accurately capture the mutual influence between processes or become computationally prohibitive in most real-world events involving large dimensions [13], [14].

Incorporating geometric structure in the form of graphs into Hawkes processes is an effective and efficient way for improving model prediction accuracy. In many real world applications, correlations between different Hawkes processes can be encoded by a graph. For example, in modeling the sequences of user-item interactions, the similarity of users and items can be represented by a user graph and an item graph, respectively. Such additional graph information can be used to impose smoothness priors on the parameters such as the base intensities of each individual process. Recently, geometric deep learning [15], [16], [17], [18] are promising techniques that can learn meaningful representations for geometric structure data such as graphs and have been successfully applied in various applications such as matrix completion.

In this paper, we propose a novel Geometric Hawkes Process (GHP) model by integrating geometric deep learning into Hawkes processes, which aims to efficiently capture meaningful patterns in a large collection of correlated sequences of recurrent events. Specifically, each sequence is modeled as a Hawkes process and the proximities between different processes are encoded in a graph. A novel convolutional and recurrent neural network is adopted to extract local meaningful patterns from the graph. The learned meaningful embeddings are then used to generate parameters such as the base intensities that characterize Hawkes processes. Comparing to traditional methods, our GHP correlates each individual Hawkes process effectively through graph embedding and it is computational efficient since the deep network structure requires constant parameters that are independent of the graph size. To the best of our knowledge, our GHP model is the first one to learn Hawkes processes with geometric deep learning. We also present the detail design of the single-graph and multi-graph cases for our Geometric Hawkes Process (GHP) model. Extensive experiments on real-world datasets demonstrate the predicting performance improvements of our model in comparison with the state of the art.

RELATED WORK

Variations of Hawkes processes have been proposed for modeling correlated sequences. For example, the work by

J. Shang and M. Sun are with the Division of Computer Science and Engineering, Louisiana State University, Baton Rouge, LA, 70803 USA e-mail: jshang2@lsu.edu, msun@csc.lsu.edu.

Manuscript received 2019; revised 2019.

Zhou et al. [3] uses a multi-dimensional Hawkes process to learn the social interactivity in a sparse low-rank network. The work by Farajtabar et al. [8] uses a Multivariate Hawkes process to model social events, which can capture both endogenous and exogenous event intensities. For modeling collections of user-item interactions, Du et al. [6] assume that sequences of all user-item pairs are independent and the coefficients of all these point processes have low rank structure. A co-evolutionary latent feature process [19] has been further proposed to construct interdependent Hawkes processes by taking advantage of additional features such as user features, item features, and interaction features between users and items. Those features are globally embedded to Hawkes processes. However, those techniques do not fully exploit the local geometric structures of different processes in the form of graphs.

Recently, geometric deep learning becomes promising because the convolutional framework can be applied on non-Euclidean data, e.g, graphs, to extract important features. Some studies such as [20] focus on the vertex domain. However, it's hard to define an appropriate neighbor for each vertex and the extracted features sometimes are not representative especially on high dimensional data structure. Another way is to formulate graph convolutional on spectral domain [15], [16], [17], [18], which is the key concept underlying our work.

The first version of Graph Convolutional Network (GCN) [15] contains n convolutional kernel parameters, which is not only computationally prohibitive but also lacks spatial localization. To solve these problems, ChebyNet [16] uses Chebyshev polynomial localized filters to replace the diagonal matrix, which reduces the computation complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Based on this type of framework, a lot of studies [17], [18] apply GCN to several specific tasks such as text classification, traffic forecasting, and matrix completion. The most closest ones to ours are the applications on matrix completion. However, their work mainly focus on modeling two dimensional data in the from of a M by N matrix without considering temporal dynamics.

TABLE I
KEY NOTATIONS.

Variable	Description
G_r, G_c	row graph and column graph
m, n	number of nodes in G_r, G_c
u, i	the $u-th$ and $i-th$ node
$\mathcal{T}$	a list of discrete temporal events
$\mathcal{O}$	observed sequences of all vertices
P	mini-batch vertices size
t_i	i-th event in $\mathcal{T}$
$\lambda(t)$	Hawkes process intensity function
$\lambda_{(u)}(t)$	Hawkes process intensity for node u
$\lambda_{(u,i)}(t)$	Hawkes process intensity for node pair (u, i)
$\kappa(t)$	kernel function in Hawkes process
a, b, c, p, q	parameters in different kernels
η	base intensity in Hawkes process
α	self-exciting coefficient in Hawkes process
$\boldsymbol{h}_u$	node u's entry of base intensity vector
$\boldsymbol{a}_u$	node u's entry of self-exciting coefficient vector
$\boldsymbol{H}_{u,i}$	node pair (u, i)'s entry of base intensity matrix
$\boldsymbol{A}_{u,i}$	node pair (u, i)'s entry of self-exciting coefficient matrix
L	Laplacian matrix of graph
D	degree matrix of graph
W	adjacency matrix of graph
Λ	diagonal eigenvalue matrix
λ_l	l-th eigenvalue in Λ
Γ	total time in all the sequences
$\tilde{\Lambda}$	scaled eigenvalues in interval $[-1, 1]$
$\tilde{L}$	rescaled Laplacian w.r.t. $\tilde{\Lambda}$
K	total degrees of Chebyshev polynomial basis
T_k	k-th degree of Chebyshev polynomial basis
$\boldsymbol{\theta}$	polynomial coefficients in single GCN
$\boldsymbol{\theta}_k$	k-th polynomial coefficient in $\boldsymbol{\theta}$
$\boldsymbol{\Theta}$	polynomial coefficients in multi GCN
$\boldsymbol{x}$	single channel input $[\boldsymbol{h}; \boldsymbol{a}]$ of single-graph GHP
$\boldsymbol{X}$	single channel input $[\boldsymbol{H}; \boldsymbol{A}]$ of multi-graph GHP
C, C'	channels of input and output
ρ, γ, β	trade-off factors for constraints
$\boldsymbol{\zeta}$	parameters in LSTM network
$\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}$	parameters in single-graph GHP
${\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)}$	the T-th step of $\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}$ in optimization
$\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}$	parameters in multi-graph GHP
${\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}}^{(T)}$	the T-th step of ${\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}}^{(T)}$ in optimization

MODEL

In this section, we introduce our Geometric Hawkes Process model. We first introduce the background of Hawkes processes and geometric deep learning, and then present the geometric Hawkes processes. We list key notations in Table I.

Background on Hawkes Processes

A univariate Hawkes process is a self-exiting temporal point process and the realization of the process consists of a list of discrete temporal events $\mathcal{T} = \{t_i\}_{i=1}^n$. It is suitable for modeling the mutual excitation between events such as the occurrences of earthquakes at a particular location. The conditional intensity function that characterizes a Hawkes process is defined as:

$$\lambda(t) = \eta + \alpha \sum_{t_i \in \mathcal{T}_t} \kappa(t - t_i), \tag{1}$$

where $\kappa(t)$ is a kernel function that captures temporal dependencies, $\eta \geq 0$ is the baseline intensity that captures the long-term incentive to generate events, $\alpha \geq 0$ is the coefficient that scales the influence of each previous event, and $\mathcal{T}_t = \{t_i | t_i < t\}_{i=1}^n$ denotes the history up to but not including time t.

Different types of parametric kernels can be used to capture certain forms of temporal dependencies for Hawkes process. For example, zero kernel assumes no decay with respect to time and the intensity with zero kernel indicates a Poisson process. A linear kernel assumes constant rate of decay with respect to time. Note that an intensity function using a linear kernel can be updated more efficiently to incorporate new events based on the accumulated value of previous events. Others complex kernels such as exponential and Rayleigh kernels assume different degrees of time decay. The specific forms of kernel functions are listed as following:

Zero Hawkes Kernel (Zero()):

$$\kappa(t) = Zero() = 0, \tag{2}$$

Linear (Linear(a,b)):

$$\kappa(t) = Linear(a, b) = a(1 - \frac{b}{a}t) \tag{3}$$

Exponential (EXP(a, b)): The exponential kernel, which is the most widely adopted by Hawkes process, is defined as:

$$\kappa(t) = EXP(a, b) = ae^{-bt}. \tag{4}$$

Power-Law (PWL(a, c, p): The power-low kernel is usually used for modeling a slower rate of decay than exponential [21]:

$$\kappa(t) = PWL(a, c, p) = \frac{a}{(t + c)^p}. \tag{5}$$

Tsallis Q-Exponential (Qexp(a, q)): The Tsallis Q-exponential kernel is a power transform along the shape parameter q between exponential and power-law kernels. It models the decay in a more hybrid way [22]:

$$\kappa(t)=Qexp(a,q)=\begin{cases} ae^{-t}, & q = 1 \\ a[1 + (q-1)t]^{\frac{1}{1-q}}, & q \neq 0 \text{ and } 1+(1-q)t > 0 \\ 0, & q \neq 0 \text{ and } 1+(1-q)t \leq 0. \end{cases} \tag{6}$$

Rayleigh (Ray(a, b)): The Rayleigh kernel has been used for modeling a non-monotonically decaying effect [23]:

$$\kappa(t) = Ray(a, b) = ate^{-bt^2} \tag{7}$$

Generally in real world applications, we would like to model a collection of correlated event sequences such as earthquakes at N locations. Intuitively, each of the N sequences can be modeled as a self-exciting Hawkes process:

$$\lambda_u(t) = \boldsymbol{h}_u + \boldsymbol{a}_u \sum_{t_j^u \in \mathcal{T}_t^u} \kappa(t - t_j^u), \tag{8}$$

where $u = 1, ..., N$ is the index of sequences such as u^{th} location, $\boldsymbol{h}$ and $\boldsymbol{a}$ are both vectors of size N and their u^{th} entries represent the non-negative base intensity and the self-exciting coefficient for u^{th} process respectively. The sequence $\mathcal{T}_t^u = \{t_j^u | t_j^u < t\}_{j=1}^n$ denotes the set of historic events of u^{th} process up to but not including time t.

For events involving a pair of entities such as the interaction events between M users and N items (e.g., various infectious diseases among a group of people), the occurrences of interaction events between user u and item i can be modeled as following:

$$\lambda_{(u,i)}(t) = \boldsymbol{H}_{u,i} + \boldsymbol{A}_{u,i} \sum_{t_j^{u,i} \in \mathcal{T}_t^{u,i}} \kappa(t - t_j^{u,i}), \tag{9}$$

where $\boldsymbol{H}$ denotes an $m \times n$ matrix with the $(u, i)^{th}$ entry equal to the non-negative base intensity for pair (u, i), $\boldsymbol{A}$ denotes an $m \times n$ matrix with the $(u, i)^{th}$ entry equal to the self-exciting coefficient for pair (u, i), and the sequence $\mathcal{T}_t^{u,i} = \{t_j^{u,i} | t_j^{u,i} < t\}_{j=1}^n$ denotes the set of historic events of pair (u, i) up to but not including time t.

However, treating each process independently would fail to achieve good performance when there are insufficient observations for each process. Incorporating correlations between processes such as location proximities and user/item similarities can improve the model prediction accuracy. The proximity between multiple Hawkes processes can be represented as an undirected weighted graph such as a proximity network of locations, a social network of users, and a network encoding item similarities.

Background on Geometric Deep Learning

Formally, an undirected weighted graph is denoted as $G = (V, E, W)$, where V is a finite set of $|V| = n$ vertices, E is the set of edges and $W \in \mathbb{R}^{n \times n}$ is the adjacency matrix with entries $W_{ij} > 0$ if $(i, j) \in E$. For each graph, a Laplacian matrix, which is an $n \times n$ symmetric positive-semidefinite matrix, can be constructed to reflect useful properties of a graph. Usually, the graph Laplacian is constructed as three different forms, the combinatorial Laplacianeq. (10), the random walk normalized Laplacianeq. (11), and the symmetric normalized Laplacianeq. (12):

$$L^c = D - W, \tag{10}$$

$$L^{rw} = D^{-1} L^c \tag{11}$$

$$L^{sys} = D^{-1/2} L^c D^{-1/2} = I_n - D^{-1/2} W D^{-1/2}, \tag{12}$$

where $D \in \mathbb{R}^{n \times n}$ is the degree matrix with $D_{ii} = \sum_j W_{ij}$ and I_n is the identity matrix. The symmetric normalized Laplacian is one of the most widely used graph Laplacian matrices. In our work, we adapt $L = L^{sys}$ as the graph Laplacian.

Graph Convolution Network (GCN): Graph convolution is typically formulated in the spectral domain through graph Fourier transform [24]. Specifically, a graph Laplacian L admits a spectral eigendecomposition of the form $L = U \Lambda U^\top$, where $U = [u_0, ..., u_{n-1}] \in \mathbb{R}^{n \times n}$ is the orthonormal matrix and is the complete set of the orthonnormal eigenvectors $\{u_l\}_{l=0}^{n-1} \in \mathbb{R}^n$, and $\Lambda = diag([\lambda_0, ..., \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$ is the diagonal matrix with the associated ordered real nonnegative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$. In particular, eigenvectors are known as the Fourier atoms in classical harmonic analysis and eigenvalues are usually interpreted as the frequencies of the graph. Given a function $\boldsymbol{x} = (x_0, ..., x_{n-1})^\top \in \mathbb{R}^n$ on the vertices of the graph, the graph Fourier transform on graph G is defined as $\hat{\boldsymbol{x}} = (\hat{x}(\lambda_0), ..., \hat{x}(\lambda_{n-1})) = U^\top \boldsymbol{x} \in \mathbb{R}^n$ and its inverse is $\boldsymbol{x} = U \hat{\boldsymbol{x}}$ [25]. Thus, the spectral convolutional of function $\boldsymbol{x}$ and convolutional kernel function $\boldsymbol{y}$ on graph G is given by [15]:

$$(\boldsymbol{x} \star \boldsymbol{y})_G = U \cdot diag([\hat{y}(\lambda_0), ..., \hat{y}(\lambda_{n-1})]) \cdot U^\top \boldsymbol{x}, \tag{13}$$

where $\odot$ is the element-wise Hadamard product. It is worth mentioning that convolutions are by definition linear operators that diagonalize in the spectral domain, according to the definition of Discrete Fourier Transform and the Convolution Theorem [24]. Thus, a GCN layer can be defined as $\boldsymbol{x}_{output} = \sigma((\boldsymbol{x} \star \boldsymbol{y})_G)$, where $diag([\hat{y}(\lambda_0), ..., \hat{y}(\lambda_{n-1})])$ represents parameters of learnable filters in the spectral domain, and σ denotes the activation function (e.g. ReLU) which is applied on the vertex-wise function values.

In order to reduce the computational complexity and the number of the parameters, as well as adding localization which is common in graph signal processing [26], a polynomial filter was introduced by [16]. Thus, the GCN layer with one filter has the following forms: $\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{K-1} \theta_k L^k \boldsymbol{x})$, where $\boldsymbol{\theta} = \{\theta_k\}_{k=0}^{K-1}$ is a vector of polynomial coefficients for such a filter and the number of parameters is K. Note that the formula involves only the computation of the Laplacian L without the computation of its decomposition of U. Specifically, the filter can be approximated by the Chebyshev polynomial basis T_k of degree k [26], where $T_k(\tilde{\lambda}_l) = 2\tilde{\lambda}_l T_{k-1}(\tilde{\lambda}_l) - T_{k-2}(\tilde{\lambda}_l)$ is defined in a recursive way with $T_0 = 1$ and $T_1 = \tilde{\lambda}_l$. Thus, the GCN layer with one filter becomes [16]:

$$\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\boldsymbol{x}), \tag{14}$$

where $\tilde{L} = 2L/\lambda_{max} - I_n$ is the rescaled Laplacian with scaled eigenvalues $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$ in the interval $[-1, 1]$.

By applying kernel polynomial localization, the computational complexity becomes $\mathcal{O}(n)$ rather than $\mathcal{O}(n^2)$ [16], as we don't need to do eigendecomposition. Also, the parameter number is only K rather than n, and the convolutional kernel with spatial localization will benefit local feature extraction. There are some simplified variants of this filter that also achieve good performance on classification tasks [17]. For example, assuming $K = 2$ and $\lambda_{max} = 2$, we can get the first-order model as:

$$\begin{aligned} \boldsymbol{x}_{output} &= \sigma(\sum_{k=0}^{1} \theta_k T_k(L - I)\boldsymbol{x}) \\ &= \sigma((\theta_0 - \theta_1 D^{-1/2} W D^{-1/2})\boldsymbol{x}). \end{aligned} \tag{15}$$

Besides, by setting the parameter of the zero-order term and the first-order term to be specific forms $\theta = \theta_0 = -\theta_1$, we have the following single parameter model which limits the number of parameter per layers to avoid over-fitting:

$$\boldsymbol{x}_{output} = \sigma(\theta(I + D^{-1/2} W D^{-1/2})\boldsymbol{x}), \tag{16}$$

A even more simplified approximation model can be obtained through a re-normalization trick [17]:

$$\boldsymbol{x}_{output} = \sigma(\theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} \boldsymbol{x}). \tag{17}$$

where $\tilde{W} = I + W$, $\tilde{D}_{ii} = \sum_j \tilde{W}_{ij}$, and $I + D^{-1/2} W D^{-1/2} \approx \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2}$.

GCN with Multi-graph (Multi-GCN): According to the definition of multidimensional Fourier Transform, Graph Fourier Transform and GCN layers can be extended to multi-graph version [27], [18]. Given two scaled graph Laplacian (referred to single-graph convolutional layer) $L_r \in \mathbb{R}^{m\times m}$ and $L_c \in \mathbb{R}^{n\times n}$ with m vertices on the row graph G_r and n vertices on the column graph G_c, a multi-GCN layer with one filter is defined as [18]:

$$\boldsymbol{X}_{output} = \sigma(\sum_{k=0}^{K-1} \sum_{k'=0}^{K-1} \theta_{kk'} T_k(\tilde{L}_r) \boldsymbol{X} T_{k'}(\tilde{L}_c)), \tag{18}$$

where function $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is two dimensional and such filter is parameterized by a $K \times K$ matrix of polynomial coefficients $\boldsymbol{\Theta} = (\theta_{kk'})$.

Generalized GCN Layers: More generally, considering the computation effectiveness of convolution, we give the following generalized form of GCN layers, which is an high performance GCN layer referring to [28] and convolution implementation in Caffe [29]. Given C input channels of $\{\boldsymbol{x}_c\}_{c=1}^{C}$ (a matrix of size $m \times C$) and C' output channels (output feature map size or the number of filters), the single-GCN layer has the generalized form:

$$\boldsymbol{x}_{c'_{output}} = \sigma(\sum_{c=1}^{C} \sum_{k=0}^{K-1} \theta_{kc,c'} T_k(\tilde{L}) \boldsymbol{x}_c). \tag{19}$$

where $c' = 1, ..., C'$.

Similarly, this can also be applied to multi-GCN layer. Given C input channels of $\{\boldsymbol{X}_c\}_{c=1}^{C}$ (a tensor of size $m \times n \times C$) and C' output channels, the multi-GCN layer has the generalized form:

$$\boldsymbol{X}_{c'_{output}} = \sigma(\sum_{c=1}^{C} \sum_{k=0}^{K-1} \sum_{k'=0}^{K-1} \theta_{kk'c,c'} T_k(\tilde{L}_r) \boldsymbol{X}_c T_{k'}(\tilde{L}_c)). \tag{20}$$

It is straightforward to expand the eqs. (15) to (17) to the generalized multi-GCN layer version. For example, for re-normalization trick model eq. (17):

$$\boldsymbol{X}_{c'_{output}} = \sigma(\sum_{c=1}^{C} \theta_{c,c'} \tilde{D}_r^{-1/2} \tilde{W}_r \tilde{D}_r^{-1/2} \boldsymbol{X}_c \tilde{D}_c^{-1/2} \tilde{W}_c \tilde{D}_c^{-1/2}), \tag{21}$$

where $\boldsymbol{\Theta} = (\theta_{c,c'})$ is the convolutional filters.

Integration of GCN and RNN: Furthermore, a GCN network coupled with a RNN network can progressively reconstruct the parameters and it has demonstrated to be highly efficient [18]. Specifically, the input of the GCN network is the original matrix $\boldsymbol{X}^{(0)}$. The output of the GCN network such as C' matrices are the input to a RNN network such as LSTM [30]. Then, the output of the RNN network are the input to a fully connected layer to calculate the changes $\boldsymbol{dX}$ of the input matrix $\boldsymbol{X}$. After several iterations (e.g. T steps), the predicted value becomes $\boldsymbol{X}^{(T)} = \boldsymbol{X}^{(T-1)} + \boldsymbol{dX}^{(T-1)}$.

Our Geometric Hawkes Processes (GHP)

We propose a novel Geometric Hawkes Process (GHP) model by integrating the geometric deep learning into Hawkes processes, which aims to efficiently capture meaningful patterns in a large collection of correlated sequences of recurrent events. In our framework, each sequence is modeled as a Hawkes process and the proximities between different processes are encoded in graphs. Specifically, we propose two types of GHP: single-graph GHP and multi-graph GHP. Single-graph GHP is particularly useful for modeling sequences with one type of graph such as modeling earthquakes at N locations with a proximity network of locations. Multi-graph GHP is particularly useful for modeling sequences with multiple graphs such as modeling the diffusion of various infectious diseases among a group of people, where the relationship of people and diseases can be represented by a user graph and an item graph, respectively. The learned meaningful embeddings from graphs are then used to generate parameters such as the base intensities that characterize Hawkes processes.

Specifically, the parameters of single-graph GHP are $\boldsymbol{h}$, $\boldsymbol{a}$ as described in eq. (8) and they are functions defined on a graph, e.g., a user graph. Similarly, the parameters of multi-graph GHP are $\boldsymbol{H}$ and $\boldsymbol{A}$ as described in eq. (9), and they are functions defined on multiple graphs, e.g., a user graph and an item graph. The parameters are random initialized as $\boldsymbol{x}$ or $\boldsymbol{X}$ in equations eq. (14) and eq. (18) respectively, and will be optimized in deep geometric learning. The loss function is defined as the log-*likelihood* of observing the sequences of events. Formally, based on the survival analysis theory [31], the *likelihood* of observing a sequence of events $\mathcal{T} = \{t_i\}_{i=1}^{n}$ is $\prod_{t_i \in \mathcal{T}} \lambda(t_i) \cdot \exp(-\int_0^{\Gamma} \lambda(\tau) d(\tau))$, where Γ is the total observation time. We present the details for the two types of GHP as the following.

Single-graph GHP: Specifically, for a collection of Hawkes processes according to eq. (8) and eq. (14), let $\mathcal{T}^u$ be the set of events induced by vertex $u = 1, ..., m$. The log-*likelihood* of observing each sequence $\mathcal{T}^u$ is:

$$\mathcal{L}(\mathcal{T}^u \mid \boldsymbol{x_{\theta,\zeta}}^{(T)}) = \sum_{t_j^u \in \mathcal{T}^u} \log(\boldsymbol{x}_u^{(T)} \Phi_j^u) - \boldsymbol{x}_u^{(T)} \Psi^u, \tag{22}$$

where:

$$\begin{aligned} \boldsymbol{x}_u^{(T)} =& (\boldsymbol{h}(u)^{(T)}, \boldsymbol{a}(u)^{(T)}), \\ \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} \kappa(t_j^u - t_k^u))^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \int_{t_j^u}^{\Gamma} \kappa(t - t_j^u) dt)^\top. \end{aligned} \tag{23}$$

The feature vector Φ_j^u and the integral Ψ^u can be pre-calculated given certain forms of kernels $\kappa(t)$. The formulas for zero kernel and linear kernel functions are straightforward. Since the constant scale parameter can be merged into the Hawkes self-exiting coefficient α (with matrix form $\boldsymbol{a}(u)^{(T)}$ and $\boldsymbol{A}(u,i)^{(T)}$) in eqs. (1), (8) and (9), we set $a = 1$ in eqs. (3) to (7). When adopting zero kernels, the second term of the feature vector Φ_j^u and the integral Ψ^u becomes zero, by integrating eq. (2) into eq. (23):

$$\begin{aligned} \Phi_j^u =& (1, 0)^\top, \\ \Psi^u =& (\Gamma, 0)^\top, \end{aligned} \tag{24}$$

When adopting linear kernels, the vectors can be computed by integrating eq. (3) into eq. (23):

$$\begin{aligned} \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} [1 - b(t_j^u - t_k^u)])^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} [\frac{b}{2}(\Gamma - t_j^u)^2 - (\Gamma - t_j^u)])^\top, \end{aligned} \tag{25}$$

For exponential kernels, the vectors can be computed by integrating eq. (4) into eq. (23):

$$\begin{aligned} \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} e^{-b(t_j^u - t_k^u)})^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{b}(1 - e^{-b(\Gamma - t_j^u)}))^\top, \end{aligned} \tag{26}$$

For power-law kernels, the vectors can be computed by integrating eq. (5) into eq. (23):

$$\begin{aligned} \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} \frac{1}{(t_j^u - t_k^u + c)^p})^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{p-1} [c^{1-p} - (\Gamma - t_j^u + c)^{1-p}])^\top, \end{aligned} \tag{27}$$

For Tsallis Q-exponential kernels with $1 < q < 2$, the vectors can be computed by integrating integrating eq. (6) into eq. (23):

$$\begin{aligned} \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} [1 + (q-1)(t_j^u - t_k^u)]^{\frac{1}{1-q}})^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{2-q} \{1 - [1 + (q-1)(\Gamma - t_j^u)]^{\frac{2-q}{1-q}}\})^\top, \end{aligned} \tag{28}$$

For Rayleigh kernels, the vectors can be computed by integrating integrating eq. (7) into eq. (23)

$$\begin{aligned} \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} (t_j^u - t_k^u) e^{-b(t_j^u - t_k^u)^2})^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{2b}(1 - e^{-b(\Gamma - t_j^u)^2}))^\top, \end{aligned} \tag{29}$$

Algorithm 1: Algorithm for Learning single-graph GHP

Input: All the training events $\mathcal{O} = \{\mathcal{T}^u\}_u$; parameters ρ, γ, β; $\{\boldsymbol{x}_c = [\boldsymbol{h}_c; \boldsymbol{a}_c]\}_{c=1}^{C}$

Output: The coefficients of Hawkes processes $\{\boldsymbol{x}_c^{(T)}\}_{c=1}^{C}$

begin

Initialize $\{\boldsymbol{x}_c^{(0)}\}_{c=1}^{C}$.

for $t \leftarrow 0$ ***to*** T **do**

Forward Propagation:

1. Apply one single-GCN layer eq. (19) on $\{\boldsymbol{x}_c^{(t)}\}_{c=1}^{C}$ producing C' output matrix $\{\boldsymbol{x}_{c'_{output}}^{(t)}\}_{c'=1}^{C'}$
2. Apply LSTM with a fully connected layer on the output matrix $\{\boldsymbol{x}_{c'_{output}}^{(t)}\}_{c'=1}^{C'}$ producing small incremental update $\{\boldsymbol{dx}_c^{(0)}\}_{c=1}^{C}$
3. Update $\{\boldsymbol{x}_c^{(t+1)} \leftarrow \boldsymbol{x}_c^{(t)} + \boldsymbol{dx}_c^{(t)}\}_{c=1}^{C}$

Back Propagation:

1. Clip Value ($\{\boldsymbol{x}_c^{(t+1)}\}_{c=1}^{C}$)
2. Apply Adam stochastic optimization algorithm to optimize eq. (30) and update weights $\boldsymbol{\theta}$, $\boldsymbol{\zeta}$

end

Output $\{\boldsymbol{x}_c^{(T)}\}_{c=1}^{C}$ to calculating Hawkes intensity by eq. (8).

end

It is worth mentioning that the notation $\boldsymbol{x_{\theta,\zeta}}^{(T)}$ emphasize the matrix depends on the parameters of GCN (polynomial coefficients $\boldsymbol{\theta}$) and those of the LSTM network (denote as $\boldsymbol{\zeta}$) after T steps. As a result, the log-*likelihood* of observing all vertices' sequences $\mathcal{O} = \{\mathcal{T}^u\}_u$ is a summation of terms by $\mathcal{L}(\mathcal{O}) = \sum_{\mathcal{T}^u \in \mathcal{O}} \mathcal{L}(\mathcal{T}^u)$. Also, we want the variables $\boldsymbol{h}$ and $\boldsymbol{a}$ to be faithful to the graph structure G with m vertices and the corresponding graph Laplacian $L_{m \times m}$. Thus, we can add the graph regularizer $h(\boldsymbol{x_{\theta,\zeta}}) = \rho\{tr(\boldsymbol{h}^\top L \boldsymbol{h}) + tr(\boldsymbol{a}^\top L \boldsymbol{a})\}$ and the squared Frobenius norm $g(\boldsymbol{x_{\theta,\zeta}}) = \gamma \|\boldsymbol{h}\|_F^2 + \beta \|\boldsymbol{a}\|_F^2$ as [32]. Finally, we can obtain $\boldsymbol{h}$ and $\boldsymbol{a}$ by minimizing the following objective function:

$$\begin{aligned} OPT =& \min_{\boldsymbol{\theta},\boldsymbol{\zeta}} - \frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^u \in \mathcal{O}} \mathcal{L}(\mathcal{T}^u \mid \boldsymbol{x_{\theta,\zeta}}^{(T)}) + h(\boldsymbol{x_{\theta,\zeta}}^{(T)}) + g(\boldsymbol{x_{\theta,\zeta}}^{(T)}) \\ & s.t. \ \boldsymbol{x_{\theta,\zeta}}^{(T)} \geq \boldsymbol{0}, \end{aligned} \tag{30}$$

where $\boldsymbol{x_{\theta,\zeta}} = [\boldsymbol{h}; \boldsymbol{a}]$, and ρ γ, β control the trade-off between the constrains. After the parameters converging to optimal, we can directly use $\boldsymbol{x}$ and eq. (8) to compute the intensity and make predictions.

Multi-graph GHP: Similarly, we can give the objective function of multi-graph GHP. According to eq. (9) and eq. (18), let $\mathcal{T}^{u,i}$ be the set of events induced between vertex $u = 1, ..., m$ and vertex $i = 1, ..., n$. The log-*likelihood* of observing each sequence $\mathcal{T}^{u,i}$ is:

$$\mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X_{\Theta,\zeta}}^{(T)}) = \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}_{u,i}^{(T)} \Phi_j^{u,i}) - \boldsymbol{X}_{u,i}^{(T)} \Psi^{u,i}, \tag{31}$$

where:

$$\begin{aligned} \boldsymbol{X}_{u,i}^{(T)} =& (\boldsymbol{H}(u,i)^{(T)}, \boldsymbol{A}(u,i)^{(T)}), \\ \Phi_j^{u,i} =& (1, \sum_{t_k^{u,i} < t_j^{u,i}} \kappa(t_j^{u,i} - t_k^{u,i}))^\top, \\ \Psi^{u,i} =& (\Gamma, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{\Gamma} \kappa(t - t_j^{u,i}) dt)^\top. \end{aligned} \tag{32}$$

Note that the feature vector $\Phi_j^{u,i}$ and the integral $\Psi^{u,i}$ can be calculated in a way similar to eq. (23). Thus, we omitted the closed forms of different kernels.

In multi-graph case, the notation $\boldsymbol{X_{\Theta,\zeta}}^{(T)}$ emphasize the matrix depends on the parameters of multi-GCN (polynomial coefficients $\boldsymbol{\Theta}$) and those of the LSTM network (denote as $\boldsymbol{\zeta}$) after T steps. Similarly, the log-*likelihood* of observing all vertices' sequences $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$ is a summation of terms by $\mathcal{L}(\mathcal{O}) = \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i})$. Given the row graph structure G_r with m vertices and the column graph structure G_c with n vertices, the corresponding graph Laplacian are $L_r \in \mathbb{R}^{m \times m}$ and $L_c \in \mathbb{R}^{n \times n}$. Thus, we can add the multi-graph regularizers as $\tilde{h}(\boldsymbol{X_{\Theta,\zeta}}) = \rho\{tr(\boldsymbol{H}^\top L_r \boldsymbol{H}) + tr(\boldsymbol{H} L_c \boldsymbol{H}^\top) + tr(\boldsymbol{A}^\top L_r \boldsymbol{A}) + tr(\boldsymbol{A} L_c \boldsymbol{A}^\top)\}$ [33]. It is worth mentioning that two matrix with $m \times n$ dimension contain too many parameters. Usually, a lot of points' attributes can be categorized into a limited number of types for the real world data. So, we assume $\boldsymbol{H}$ and $\boldsymbol{A}$ have low-rank structures, and we can add the nuclear norm $\tilde{g}(\boldsymbol{X_{\Theta,\zeta}}) = \gamma \|\boldsymbol{H}\|_* + \beta \|\boldsymbol{A}\|_*$ as [6], which is frequently used as a convex surrogate penalty term for matrix rank. Finally, we can obtain $\boldsymbol{H}$ and $\boldsymbol{A}$ by minimizing the following objective function:

$$\begin{aligned} OPT = & \min_{\boldsymbol{\Theta},\boldsymbol{\zeta}} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} | \boldsymbol{X_{\Theta,\zeta}}^{(T)}) + \tilde{h}(\boldsymbol{X_{\Theta,\zeta}}^{(T)}) + \tilde{g}(\boldsymbol{X_{\Theta,\zeta}}^{(T)}) \\ & s.t.\ \boldsymbol{X_{\Theta,\zeta}}^{(T)} \geq \boldsymbol{0}, \end{aligned} \tag{33}$$

where $\boldsymbol{X_{\Theta,\zeta}} = [\boldsymbol{H}; \boldsymbol{A}]$, and ρ γ, β control the trade-off between the constrains. After the parameters converging to optimal, we can directly use $\boldsymbol{X}$ and eq. (9) to compute the intensity and make predictions.

Learning with Clipping: We can use several stochastic optimization algorithms such as SGD and Adam [34] to solve the log-*likelihood* with regularizers. However, as Hawkes processes have non-negative parameters, the objective function should be optimized under such non-negative constraints eqs. (30) and (33). Since it is the inequality constraints, directly solving it by adding Lagrange multiplier or Kuhn-Tucker method [35] will introduce the Complementary Slackness Conditions, which makes it more complex. To enforce the non-negative constraints on the objective function, we clip the value to lie within a compact space after each temporal step $t = 0, ..., T$ and make the lower bound greater than zero. We present the following learning algorithms 1 and 2 for single-graph and multi-graph GHP, respectively.

Algorithm 2: Algorithm for Learning multi-graph GHP

Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; parameters ρ, γ, β; $\{\boldsymbol{X}_c = [\boldsymbol{H}_c; \boldsymbol{A}_c]\}_{c=1}^C$
Output: The coefficients of Hawkes processes $\{\boldsymbol{X}_c^{(T)}\}_{c=1}^C$
begin
  Initialize $\{\boldsymbol{X}_c^{(0)}\}_{c=1}^C$.
  for $t \leftarrow 0$ ***to*** T **do**
    Forward Propagation:
    1. Apply multi-GCN layer eq. (20) on $\{\boldsymbol{X}_c^{(t)}\}_{c=1}^C$ producing C' output matrix $\{\boldsymbol{X}_{c'_{output}}^{(t)}\}_{c'=1}^{C'}$
    2. Apply LSTM with a fully connected layer on the output matrix $\{\boldsymbol{X}_{c'_{output}}^{(t)}\}_{c'=1}^{C'}$ producing small incremental update $\{\boldsymbol{dX}_c^{(0)}\}_{c=1}^C$
    3. Update $\{\boldsymbol{X}_c^{(t+1)} \leftarrow \boldsymbol{X}_c^{(t)} + \boldsymbol{dX}_c^{(t)}\}_{c=1}^C$
    Back Propagation:
    1. Clip Value $(\{\boldsymbol{X}_c^{(t+1)}\}_{c=1}^C)$
    2. Apply Adam stochastic optimization algorithm to optimize eq. (33) and update weights $\boldsymbol{\Theta}$, $\boldsymbol{\zeta}$
  end
  Output $\{\boldsymbol{X}_c^{(T)}\}_{c=1}^C$ to calculating Hawkes intensity by eq. (9).
end

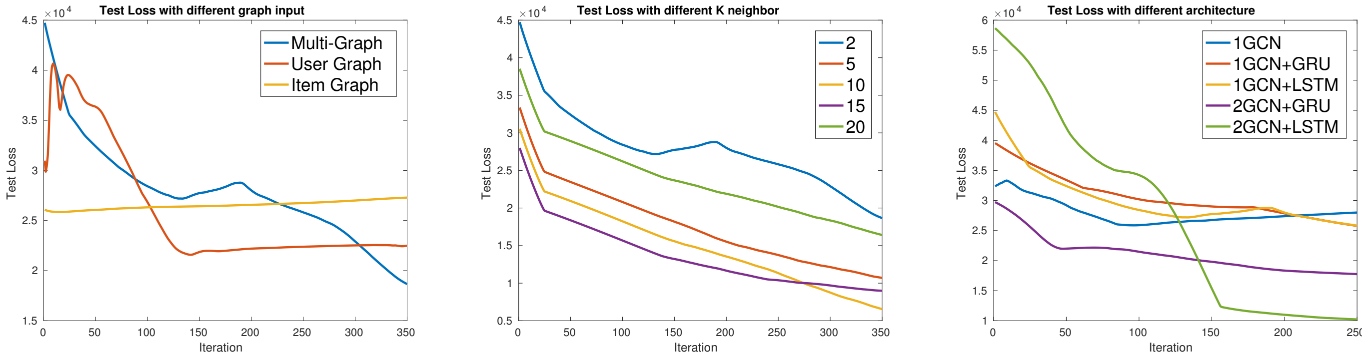
Computational Complexity: By applying polynomial localization, the single-GCN eq. (14) reaches $\mathcal{O}(n)$ [16] rather than using eq. (18) with complexity $\mathcal{O}(n^2)$, where n is the number of vertices of the graph. Thus, the multi-GCN has the complexity of $\mathcal{O}(mn)$ [18] considering $C, C', K \ll min(m,n)$. Also, the learning complexity of LSTM network is $\mathcal{O}(\boldsymbol{W})$, where the number of parameters $\boldsymbol{W} = 4n_c^2 + 4n_c n_i + n_c n_o + 3n_c$ [36], and the number of memory units, input units and output units are equal to the number of output feature map size of the GCN $n_c = n_i = n_c = C'$ in our network. As a result, such single-GCN + RNN network has the complexity of $\mathcal{O}(n + n \cdot C' \cdot C') = \mathcal{O}(n)$ per time step and the multi-graph one has the similar complexity of $\mathcal{O}(mn)$ per time step. It is worth mentioning that these are computed globally. To make it more efficient, we can also address several mini-batch with P samples from n or mn, which makes the algorithm independent of the graph size and achieve $\mathcal{O}(P)$ complexity.

EXPERIMENT AND RESULTS

In this section, we introduce the experiments.

Experimental Settings and Evaluation Metrics

We evaluate our model on three real world datasets which contain temporal interactions between a set of users and a set of items. The details are shown in table II. Specifically, the **IPTV** dataset [2] contains 7100 users and 436 TV programs with 1420 program features such as genres and countries. For each user-item pair, it contains a sequence of viewing time during the period of January to November 2012. The



(a) Graph input (b) K neighbor (c) Architecture

Fig. 1. Testing loss with respect to different graph inputs, different number of neighbors, and architectures on IPTV data.

TABLE II
DATASET DESCRIPTION.

Dataset	User	Item	Event	Pair	Item-Feature	Time
IPTV	7100	436	$2.4M$	4726	1420	8040
Yelp	100	$17K$	$35K$	20246	823	44640
Reddit	1000	1403	$10K$	2053	35	4090

Yelp[1] dataset is available from Yelp dataset challenge. After pre-processing, it records the time of writing reviews for 17k businesses by 100 users during a period of 11 years. The **Reddit**[2] dataset contains the time of posting discussions between random selected 1000 users and 1403 threads in January 2014.

As suggested in [18], a user or item graph can be constructed as an unweighted k-nearest neighbor graph in the space of features such as TV features. In cases where user and item features are not available, we can construct a two-dimensional user-item matrix from the time sequences where each entry indicates the total count of user-item interactions, and apply SVD to get a latent feature vector for each use or item. In cases where user and item content features (e.g., TV genres and countries) are available, we investigate the effect of building a KNN graph with different integration methods of content features and the SVD features obtained through user-item matrix. We can model these datasets using either single-graph GHP or multi-graph GHP. For the first case, the parameters are regraded as vector functions on a graph (e.g., user graph) and the values of each dimension (e.g., item index) are regraded as different channels. For the second case, the parameters are regraded as scalar functions on both user and item graphs and the size of the input channel is one.

There are three metrics to evaluate the performance of the model. In the experiments, we use the events before time $T \cdot p$ as the training data, and the rest of them as testing data, where T is the length of the total time, and $p = 0.76$ is the proportion where we split the data.

Test Loss: It is defined as in the objective function eqs. (30) and (33) with fixed coefficients of Hawkes processes learned using events in the training set.

Item Relevance: Given the history $\mathcal{T} = \{t_i\}_{i=1}^n$ of a specific user u, we calculate the survival rates for all the items at each testing time t, that is $S_i(t) = \exp(-\int_{t_n^i}^t \lambda_i(\tau) d(\tau))$. We then order all the survivals and compute the rank of the ground-truth item the user interacts at testing time t. Ideally the ground-truth item should be ordered at rank one. Following [19], we report *mean average rank* (MAR) of all testing cases. A smaller value of MAR indicates better predictive performance.

Time Prediction Accuracy: Given a specific pair of user u and the item i, we record the *mean absolute error* (MAE) of the next predicted time and the ground truth of testing time t. The predicted time is calculated by the density of next event time as $f(t) = \lambda_{(u,i)}(t) S_{(u,i)}(t)$, and then use the expectation to predict the next event. Furthermore, we also give the relative percentage of the prediction error (Err %).

Baseline Methods

Po: Poisson processes are simplified Hawkes processes without capturing temporal dependencies. The only parameter to characterize Poisson is the base intensity η, which is a constant.

Po-T: Poisson-Tensor uses Poisson regression error instead of RMSE as the loss function when fitting the data. The intensity are regarded as the number of events in each discretized time slot [37]. It assumes that the missing values are not random, and thus simulating the values with Poisson distribution is more reasonable than with Gaussian. Once we get the model parameters, there are two ways to simulate the intensity of test data. One is using the intensity that we have got only in the last time interval, and the other is using the average intensity of all the training time intervals. We report the best performance of these two choices.

LRH: LowRankHawkes is a collection of Hawkes processes [6] assuming that all processes are independent and the parameters are low rank matrices. However, there are no interactions between different processes.

Coevol: Coevolve is a co-evolutionary latent feature process [19] which constructs interdependent Hawkes processes by embedding user and item features globally into each process. This method actually combines all events happening

[1] https://www.yelp.com/dataset/challenge
[2] https://dynamics.cs.washington.edu/data.html

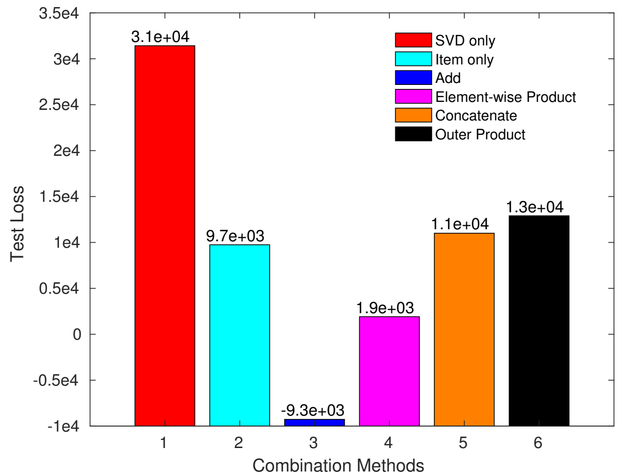


Fig. 2. Test loss with respect to different combination methods of item features on IPTV dataset.

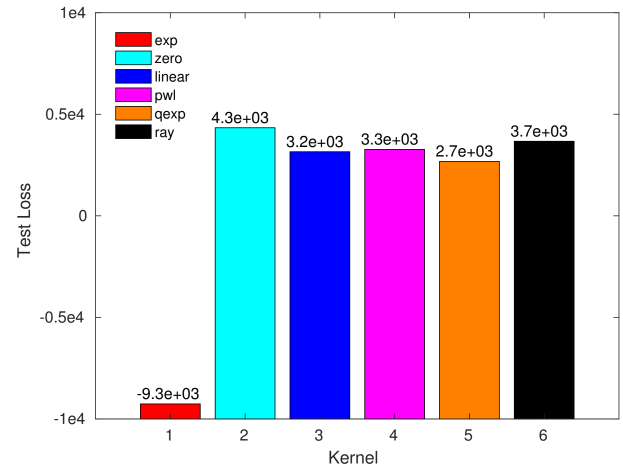


Fig. 3. Test loss with respect to different kernels on IPTV dataset.

before the current event from different processes when fitting the parameter of each individual process. However, the performance in terms of item relevance may be affected due to unrelated events. In addition, if no features are used, the model reduces to a Poisson process.

Compare Ours of Different Parameters

We first investigate the influence of important parameters in our GHP model by evaluating them using the testing loss. Specifically, the main parameters are types of graph, k-nearest neighbor, and the variations of deep learning architectures. Moreover, we consider different ways of building graphs to integrate both content features and interaction features, the type of triggering kernels, and different graph propagation models.

Single vs Multi-graph: We show the results of testing losses with multi-graph input compared with only single-graph input, e.g. only a user graph or an item graph, of IPTV dataset in fig. 1(a). As we can see, the testing loss with multi-graph input outperforms that with only single-graph input, which prove that the graph information is extracted well by the GCN + LSTM networks. It is worth mentioning that the IPTV dataset contains 7100 users and 436 TV show items, so using only the user graph achieves better results than using only the item graph. Also, the testing loss shows that the less information inputted, the faster it overfits the data.

Number of K-neighbors: We also investigate the number K's effect when constructing the K-nearest neighbor graph. In fig. 1(b), we present the testing losses of IPTV dataset with 2, 5, 10, 15, 20 -NN graph input of multi-graph GHP model. The figure demonstrates that give the K in a reasonable range, we can achieve a stable and accurate estimation of the model. The results show that $k = 10$ is the best for IPTV dataset. In the experiment, we use the same K for both user and item input graphs. However, we can separately set K for the user graph and the item graph to make it more flexible. According to fig. 1(a) and fig. 1(b), our GHP model benefits from the input graph information and extracts useful features from these interactions, and thus the model overcomes the isolation of point process models such as [6].

Variations of Architecture Setting: We compare different architecture settings of our model and the results are presented in fig. 1(c). First of all, the RNN structure such as LSTM or GRU is essential to learn the diffusion process of coefficients. The LSTM is more effective compared to GRU [38] because LSTM can remember more historical information. Besides, the results show that adding more GCN layers enhances the performance of modeling Hawkes processes, which indicates that deeper network may extract more useful features. As data size increases, it is necessary to build deeper architectures. More extensive studies on the architecture of GCN in different applications can be found at [17], [18]. In our experiment, we found the structure of two GCN layers plus one LSTM layer works best.

Building Item Graph by Integrating Features: In the case where both item content features (e.g., TV genres and countries) and user-item interaction exist, we investigate different combination methods to integrate features to construct the item KNN graph using IPTV dataset. The KNN graph depends on the distance between user and item similarities based on these two type of features. Since the item content features are quite sparse with high dimension, we first apply some dimension reduction methods on them to reduce the dimension of these features to the same dimension k of the item latent feature obtained through user-item interactions. We adopt three dimension reduction techniques: Principal Component Analysis (PCA), Auto Encoder(AE), and Multi-dimensional Scaling (MDS). The experiments show that PCA achieves the best performance, while MDS is the second and AE becomes the worst. After normalizing these two types of features, we finally integrate them to a unified feature vector by six methods: only item SVD features by factoring user-item interaction; only item content features; element-wise addition of these two features; element-wise product of the two; concatenation of the two that extend the low dimension from k to $2k$; and the outer product which extend the lower dimension from k to k^2.

We present the performance of test loss with respect to different combination methods of item features on IPTV dataset

TABLE III
TEST LOSS WITH RESPECT TO DIFFERENT GRAPH PROPAGATION MODELS ON IPTV DATASET.

Description	Propagation Model	Test Loss
$K = 2$ Chebyshev filter eq. (14)	$\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\boldsymbol{x})$	-9.26e+03
$K = 3$ Chebyshev filter eq. (14)		-7.11e+03
1^{st}-order model eq. (15)	$\boldsymbol{x}_{output} = \sigma((\theta_0 - \theta_1 D^{-1/2} W D^{-1/2})\boldsymbol{x})$	-5.04e+03
Single parameter 1^{st}-order model eq. (16)	$\boldsymbol{x}_{output} = \sigma(\theta(I + D^{-1/2} W D^{-1/2})\boldsymbol{x})$	-3.14e+03
Re-normalization trick eq. (17)	$\boldsymbol{x}_{output} = \sigma(\theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2}\boldsymbol{x})$	-2.37e+03
1^{st}-order term only	$\boldsymbol{x}_{output} = \sigma(\theta D^{-1/2} W D^{-1/2}\boldsymbol{x})$	-9.69e+03

in fig. 2. First of all, adopting the integration of two types of features is better than only adopting one type of features. Second, the item content features seem to have better qualities in representations than SVD collaborative features. At last, it seems that addition and element-wise product operations of these two type of features achieve better performance than others. The redundant and noisy information generated by concatenation and outer product operations seem to be the reason leading worse performance.

Effect of Triggering Kernels: We also investigate the effects of using different triggering kernels of Hawkes processes such as zero kernels, linear kernels and some other kernels introduced before eqs. (4) to (7). As shown in fig. 3, the most widely used exponential kernel seems to capture the dependence of history events well and achieves the best performance. Zero kernel is the worst and others are in-between. Some research indicates that these kernels, which represent different forms of decay may perform differently depending on various types of data [23], [4].

Different Graph Propagation Models: We compare the test loss with different graph propagation models [17] on the IPTV dataset. The results shown in table III indicated that the first order term only model and the Chebyshev filter eq. (14) with $K = 2$ both achieve comparable performance than other graph propagation models eqs. (15) to (17). As we increase K, the number of parameters increases, which may lead to the over-fitting problem. In our experiments, these graph propagation models don't effect the performance too much comparing with choices of kernel and feature combination methods. This indicates that the content of the graph seems to be more important than how it is embedded into the learning under the framework of graph convolution networks. Based on all the experiments, we conclude that the graph Laplacian which depends on the graph construction and the graph convolution network structures are crucial to the performance of our GHP model.

Compare with Baselines

We compare our GHP model with some state-of-art baselines by evaluating the metrics of item relevance and time prediction accuracy as shown in table IV. We use multi-graph GHP model and the results show that our method outperforms other baseline methods in general. For IPTV and Reddit datasets, the exception occurs on time prediction of *Coevol*. Specifically, the *Coevol* method uses a weighted summation of all the events happened before the current event to simulate one point's intensity. Therefore, the returning-time prediction is good since a large number of events are used to simulate the intensity function. The embedding of auxiliary features such as TV genres is also helpful in improving prediction accuracy. However, the item relevance prediction becomes worse [19] because the parameters of the individual process are influenced by unrelated processes. Meanwhile, we can see that the Hawkes process based models, such as our model, *Coevol*, and *LRH*, get better performances when there are sufficient history events (with nearly 400 events per point for IPTV and 30 events for Reddit) in comparison with the Poisson related models. For Yelp data, as each point process only has fewer than 3 events in average, the time prediction is similar among *LRH* and *Po*, which means that the history is not such an important factor. In this time sparsity case, factorization model *Po-T* gets better results than point process based models. For all three datasets, *LRH* with low-rank assumption, performs worse than our GHP that integrates graphs with low rank assumption. Obviously, integrating graphs can better capture the correlations between different processes.

TABLE IV
AVERAGE PREDICTION PERFORMANCE COMPARISON ON IPTV, YELP, AND REDDIT DATASETS.

Datasets	Metrics	Methods				
		Our	LRH	Coevol	Po	Po-T
IPTV	MAR	**1.643**	5.175	13.57	173.7	178.7
	MAE	361.0	822.1	**160.3**	993.1	933.6
	Err %	5.13	12.27	**2.35**	14.83	13.89
Yelp	MAR	**94.62**	116.0	671.2	7778	1738
	MAE	**499.0**	845.7	587.3	850.9	587.1
	Err %	**14.59**	23.71	17.49	23.91	17.48
Reddit	MAR	**6.010**	49.14	82.44	128.2	85.49
	MAE	5367	8476	**5323**	10314	9155
	Err %	**14.15**	21.50	14.27	26.59	24.09

CONCLUSIONS

In this paper, we present a novel framework that integrates the graph convolutional recurrent neural network and Hawkes processes to model temporal events. Our model can be applied to a collection of correlated temporal sequences of recurrent events, and it is able to correlate each sequence through graph embedding. We also present single-graph and multi-graph settings of our model. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art. Future work includes integrating Hawkes Processes with other different types of deep neural network structures and extending to other applications.

REFERENCES

[1] H. Xu, W. Wu, S. Nemati, and H. Zha, "Patient flow prediction via discriminative learning of mutually-correcting processes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 157–171, 2017.

[2] H. Xu, M. Farajtabar, and H. Zha, "Learning granger causality for Hawkes processes," in *Proc. of the 33rd International Conference on Machine Learning*, 2016, pp. 1717–1726.

[3] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes," in *Proc. of the 16th International Conference on Artificial Intelligence and Statistics*, 2013, pp. 641–649.

[4] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 1555–1564.

[5] J. Shang and M. Sun, "Local low-rank hawkes processes for temporal user-item interactions," in *Proc. of the IEEE International Conference on Data Mining (ICDM)*, 2018.

[6] N. Du, Y. Wang, N. He, J. Sun, and L. Song, "Time-sensitive recommendation from recurrent user activities," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 3492–3500.

[7] T. J. Liniger, "Multivariate Hawkes processes," Ph.D. dissertation, ETH Zurich, 2009.

[8] M. Farajtabar, N. Du, M. G. Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2474–2482.

[9] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 6754–6764.

[10] Y. Yang, J. Etesami, N. He, and N. Kiyavash, "Nonparametric hawkes processes: Online estimation and generalization bounds," *arXiv preprint arXiv:1801.08273*, 2018.

[11] R. Lemonnier, K. Scaman, and A. Kalogeratos, "Multivariate Hawkes processes for large-scale inference." in *Proc. of the AAAI Conference on Artificial Intelligence*, 2017, pp. 2168–2174.

[12] J. Etesami, N. Kiyavash, K. Zhang, and K. Singhal, "Learning network of multivariate Hawkes processes: a time series approach," in *Proc. of the 32nd Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2016, pp. 162–171.

[13] M. Eichler, R. Dahlhaus, and J. Dueck, "Graphical modeling for multivariate Hawkes processes with nonparametric link functions," *Journal of Time Series Analysis*, vol. 38, no. 2, pp. 225–242, 2017.

[14] E. C. Hall and R. M. Willett, "Tracking dynamic point processes on networks," *IEEE Transactions on Information Theory*, vol. 62, no. 7, pp. 4327–4346, 2016.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 3844–3852.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[18] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 3697–3707.

[19] Y. Wang, N. Du, R. Trivedi, and L. Song, "Coevolutionary latent feature processes for continuous-time user-item interactions," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 4547–4555.

[20] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[21] E. Bacry, I. Mastromatteo, and J.-F. Muzy, "Hawkes processes in finance," *Market Microstructure and Liquidity*, vol. 1, no. 01, p. 1550005, 2015.

[22] D. Ghoshdastidar and A. Dukkipati, "On power-law kernels, corresponding reproducing kernel hilbert space and applications," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2013.

[23] M. G. Rodriguez, "Structure and dynamics of diffusion networks," Ph.D. dissertation, 2013.

[24] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[25] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[26] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[27] T. Kurokawa, T. Oki, and H. Nagao, "Multi-dimensional graph fourier transform," *arXiv preprint arXiv:1712.07811*, 2017.

[28] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] O. Aalen, O. Borgan, and H. Gjessing, *Survival and Event History Analysis: A Process Point of View*. Springer Science & Business Media, 2008.

[32] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 2107–2115.

[33] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," *arXiv preprint arXiv:1408.1717*, 2014.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[35] B. Wallace, "Constrained optimization: Kuhn-tucker conditions," 2004.

[36] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[37] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.