May 2020

Predictive Modeling of Asynchronous Event Sequence Data

Jin Shang

PREDICTIVE MODELING OF ASYNCHRONOUS EVENT SEQUENCE DATA

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Jin Shang
B.S., University of Science and Technology of China, 2013
M.S., University of Science and Technology of China, 2016
August 2020

To my Mom and Dad.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to my advisor, Dr. Mingxuan Sun, for not only continuously supporting my study and related research but also her visionary, inspiring, and insightful guidance. I would also like to thank my committee members, Dr. Konstantin Busch, Dr. Jian Zhang, Dr. Hongchao Zhang and Dr. Boryung Ju for their valuable discussions and constructive suggestions on my dissertation.

I am grateful to my colleagues: Zihan Zhou, Qing Wang, Qing Chen, Changbin Li, Fei Li, Guangyue Xu, Zhongzhu Peng, and Alimire Nabijiang for discussions about research and study. I would also like to thank Dr. Nina Lam, Zheye Wang, Kejin Wang and Volodymyr Mihunov in the Department of Environmental Sciences for active discussions and close collaborations on the research project. I have great memories and experiences at NewsBreak during my summer internship. I would like to appreciate my mentor Dr. Ke Zhou for bringing me to to many new fields, which greatly broadens the horizon of my research.

Finally, I want to thank my parents for their love and great support throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Large volumes of temporal event data, such as online check-ins and electronic records of hospital admissions, are becoming increasingly available in a wide variety of applications including healthcare analytics, smart cities, and social network analysis. Those temporal events are often asynchronous, interdependent, and exhibiting self-exciting properties. For example, in the patient's diagnosis events, the elevated risk exists for a patient that has been recently at risk. Machine learning that leverages event sequence data can improve the prediction accuracy of future events and provide valuable services. For example, in e-commerce and network traffic diagnosis, the analysis of user activities can be used to predict and control dynamic user traffic demand to improves risk response efficiency. In this work, we investigate and design novel point process-based models and learning algorithms to analyze dynamic event sequence data from various aspects. (1) We first propose local low-rank Hawkes processes to capture the mutual influences between sequences of multiple event types. (2) We then develop geometric Hawkes processes to integrate geometric structures to point processes based on graph convolutional recurrent neural networks to improve prediction accuracy. (3) We introduce several novel fairness metrics to penalize the event likelihood function in order to tackle the challenge of data bias and the amplified through self-excitation in point processes. (4) We also propose a novel list-wise fairness criterion for point processes that can efficiently evaluate the ranking fairness in event prediction, and present a strict definition of the unfairness consistency property of a fairness metric.

CHAPTER 1
INTRODUCTION

1.1 Motivation

With the fast development of modern communications, networks and mobile devices, millions of people around the world generate huge volumes of event data every day in a wide variety of domains including social networks, business directory service, job market, and TV systems. These event data are usually asynchronous, interdependent and exhibiting self-exciting properties, which contains a series of events with time stamp, marks and optionally high-dimensional features. For example, in some business directory services like Yelp, we post our feedback to the business which is likely to trigger or be triggered by other people's feedback. The feedback of a user can be viewed as a series of asynchronous interdependent events, and the content of feedback as well as the user or business profiles can be transformed into some high-dimensional features.

Modeling and analysis of these event sequence data by machine learning can improve the prediction accuracy of future events and provide valuable services, for instance, making business more profitable. However, traditional temporal analysis methods usually fail to tackle the problem of asynchronous event sequences, since they contain discrete events seemingly ”randomly” distributed in the continuous-time domain. Thus, not only the order of events but also the time intervals between them are crucial for describing the mutual influences between sequences, which increases the difficulty of modeling and analysis. Fortunately, we can achieve this goal with the help of temporal point processes theory.

1.2 Research Problems and Contributions

The main interests of our research lie in designing novel point process-based models and learning algorithms to analyze dynamic event sequence data from various aspects. Specifically, We investigate three challenging research problems:

• **User-item Interaction Modeling.** Hawkes processes have become very popular in modeling multiple recurrent user-item interaction events that exhibit mutual-excitation properties in various domains. Generally, modeling the interaction sequence of each user-item pair as an independent Hawkes process is ineffective since the prediction accuracy of future event occurrences for users and items with few observed interactions is low. On the other hand, multivariate Hawkes processes (MHPs) can be used to handle multi-dimensional random processes where different dimensions are correlated with each other. However, an MHP either fails to describe the correct mutual influence between dimensions or become computational inhibitive in most real-world events involving a large collection of users and items.

To tackle this challenge, the Chapter 2 we propose local low-rank Hawkes processes to model large-scale user-item interactions, which efficiently captures the correlations of Hawkes processes in different dimensions. In addition, we design an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art.

• **Incorporating Geometric Structure.** 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 Chapter 3, 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.

• **Balancing Performance and Fairness.** Point processes are very popular in modeling self-exciting user-item interaction events such as online check-ins, job-seeking events, and electronic records of hospital admissions. Although current studies are promising in improving event prediction accuracy, there are still critical research challenges due to data sparsity and imbalances with respect to certain event groups. Firstly, the prediction accuracy for each user-item pair may decrease when there are insufficient interaction events. Secondly, unfair predictions may be generated due to data bias and can be amplified through self-excitation.

In Chapter 4, we propose a novel graph convolutional point process framework that incorporates dynamic coevolutionary feature embedding into the geometric structure to tackle data sparsity. We also introduce several novel fairness metrics to penalize the event likelihood function to enforce fairness. These fairness metrics all enjoy convexity and permitting efficient optimization as regularizers. Extensive experiments on real-world datasets demonstrate that our method improves event prediction over baselines and controls the balance between accuracy and fairness effectively.

• **List-wise Fairness Criterion.** Many types of event sequence data exhibit triggering and clustering properties in space and time. Point processes are widely used in modeling such event data with applications such as predictive policing and disaster event forecasting. Although current algorithms can achieve significant event prediction accuracy, the historic data or the self-excitation property can introduce biased prediction. For example, hotspots ranked by event hazard rates can make the visibility of a disadvantaged group (e.g., racial minorities or the communities of lower social economic status) more apparent. Existing methods have explored ways to achieve parity between the groups by penalizing the objective function with several group fairness metrics. However, these metrics fail to measure the fairness on every prefix of the ranking.

In Chapter 5, we propose a novel list-wise fairness criterion for point processes, which can efficiently evaluate the ranking fairness in event prediction. We also present a strict definition of the unfairness consistency property of a fairness metric and prove that our list-wise fairness criterion satisfies this property. Experiments on several real-world spatial-temporal sequence datasets demonstrate the effectiveness of our list-wise fairness criterion.

1.3 Literature Survey

1.3.1 Local Low-rank Matrix Completion with Kernel Smoothing

Local low-rank matrix completion with kernel smoothing [51] has been applied to matrix factorization, in which the observed ratings are formulated as a matrix and simulated with several local mappings. Each local mapping is assumed to be low-rank and the missing ratings are reconstructed with a non-parametric regression of those mappings. Previous work [51] mainly focuses on two-dimensional matrix factorization without the temporal dimension. It enforces the low-rank assumption by explicitly decomposing a matrix to the product of two low-rank matrices. Variations of matrix completion methods [52, 53, 78] are widely applied in recommender systems.

1.3.2 Temporal Point Process

Point processes have been widely used in various applications such as predictive policing [63, 62] and predicting recurrent online user behaviors [19, 83, 89]. High dimensional intertwined stochastic processes have been introduced to capture coevolutionary dynamics in [25, 84, 79] to achieve promising performance. Specifically, additional features such as user features, item features, and user-item interaction features are embedded dynamically into the intensity of point processes. However, the prediction accuracy of future event occurrences usually decreases for users and items with very few observed interactions.

1.3.3 Hawkes Process

Hawkes processes [35] can be used in a variety of applications such as inferring granger causality [87], modeling patient records in smart health [89], and predicting online social activities [97]. For example, a multi-dimensional Hawkes process has been proposed by Zhou et al. [97] to learn the social event diffusion in sparse low-rank networks. A multivariate Hawkes process has further been proposed by Farajtabar et al. [24] to capture both endogenous and exogenous event intensities in social network events. Limitations of the multivariate Hawkes process such as the computational inefficiency for modeling real-world events have been studied in [22, 45, 32].

For modeling collections of user-item interactions, Du et al. [20] 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 [84] 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.

1.3.4 Geometric Deep Learning

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 [67] 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 [8, 18, 44, 65], which is more widely used in recently research works.

The first version of Graph Convolutional Network (GCN) [8] contains n convolutional kernel parameters, which is not only computationally prohibitive but also lacks spatial localization. To solve these problems, ChebyNet [18] 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 [44, 65] apply GCN to several specific tasks such as text classification, traffic forecasting, and matrix completion.

1.3.5 Fairness in Machine Learning.

Machine learning and artificial intelligence (AI) systems exhibit bias due to a number of factors including the human bias in training data and the design of algorithm models [3]. It is also well known that machine learning and AI algorithms may reproduce and even amplify human biases and social inequities especially in applications involving feedback loops such as predictive policing [39, 62]. There are many definitions of fairness such as group parity [9], equalized odds [34, 94], individual fairness [21], and counterfactual fairness [71]. Group parity and its variations are widely applied in classification and regression tasks [42, 92, 7].

The impact of imposing fairness constraints to machine learning and AI is dependent on the specific domain datasets, the specific fairness definition, and the prediction algorithms. Most of the models and algorithms proposed to improve fairness fall into three categories: pre-processing [96, 57], optimization at training time [42, 94, 10], and post-processing [26, 34]. Generally, training time optimization, which is domain-specific, can achieve good performance on accuracy and fairness measures and offers the flexibility to balance the trade-off between accuracy and fairness measures.

Recent studies have focused on the fairness problem of ranking. Specifically, a fairness measure is proposed in [90] to compare the distributions between two demographic groups at several prefixes with a discount factor based on an inverse logarithmic function. However, the definition of ranking fairness is heuristic with no rigorous proof and only preliminary results are demonstrated. An auditing framework is proposed [46] to measure search engine bias. The work focuses on auditing ranking algorithms to identify the sources of bias rather than generating a fair ranking list. A recent work [95] presents a ranked group fairness criterion based on the statistical hypothesis testing. The method can adjust a ranking list so that a minimal number of instances in protected groups must appear in the top-k list to guarantee a fair criterion. However, this post-processing algorithm and the training of ranking function are independent and thus the adjustment is limited. A variation of group parity is proposed in [62] for top-K crime hotspots prediction. The fairness loss is integrated into the likelihood of event occurrences and the model parameters are penalized so as to balance the trade-off between accuracy and fair loss. However, the fairness metric does not guarantee group parity at any point in the ranked list.

1.4 Organizations

The remainder of this research proposal is organized as follows. In Chapter 2, we propose local low-rank Hawkes processes to capture the mutual influences between sequences of multiple event types. In Chapter 3, we present geometric Hawkes processes to integrate geometric structures to point processes based on graph convolutional recurrent neural networks to improve prediction accuracy. Then in Chapter 4, we introduce several novel fairness metrics to penalize the event likelihood function in order to tackle the challenge of data bias and the amplified through self-excitation in point processes. Finally in Chapter 5, we propose a novel list-wise fairness criterion for point processes that can efficiently evaluate the ranking fairness in event prediction, and present a strict definition of the unfairness consistency property of a fairness metric. Chapter 6 concludes this dissertation and discusses future research directions.

CHAPTER 2
LOCAL LOW-RANK HAWKES PROCESSES

2.1 Introduction

Hawkes processes have become very popular in modeling recurrent user-item interaction events that exhibit mutual-excitation properties in various domains [97, 24]. For example, Hawkes processes can be used to model user behaviors in online services, where the interaction of a user with an item such as visiting a website or watching a movie may trigger future interactions with other correlated items. Recent approaches [20, 86] treat the event occurrences of each user-item pair as a point process and predict the next occurrence of user-item interaction based on previous interactions. Accurate modeling user-item interactions may have significant economic impact on online platforms such as revenue boost due to targeted advertising.

Formally, the Hawkes process for modeling an interaction sequence of a single user-item pair (u, i) can be characterized by parameters such as a base intensity and a self-exciting coefficient that captures the influence of each previous event. Intuitively, m-by-n Hawkes processes can be used to model interaction sequences for m users and n items, where the base intensities and the self-exciting coefficients are represented as m-by-n matrices, respectively. Since users and items can usually be grouped into a limited number of clusters, we can assume that each parameter matrix has a low-rank structure. However, the prediction accuracy of future event occurrences for users and items with few observed interactions is low since the point processes are independent of each other [20]. In fact, only a few recurrent events such as purchases are observed for a majority of pairs of users and items in many large-scale real-world scenarios.

One way to alleviate the cold-start issue is to incorporate auxiliary features such as user demographics and item content features. For example, a coevolutionary model [84] takes advantage of auxiliary features such as item genres and incorporates the former events of all user-item pairs with different weights. The time prediction performance has been improved since more data are used to fit the model parameters of each user-item pair, but the item prediction performance has decreased due to the combination of the events from all user-item pairs.

On the other hand, a multivariate Hawkes process (MHP) [35, 56] can be used to handle a multi-dimensional (e.g., $N = m \times n$) random process where different dimensions are correlated with each other. Specifically, the conditional intensity for the i-th dimension is characterized by the base intensity and the linear combination of the influences of events occurred in every other dimension on the i-th dimension. Extensive research [54, 23, 22, 87] has focused on estimating the $N \times N$ excitation matrix of a multivariate process for various inference tasks. However, an MHP either fails to describe the correct mutual influence between dimensions or becomes computational expensive in most real-world applications involving a large collection of event sequences [22, 45, 32].

In this chapter, we propose local low-rank Hawkes processes to model large-scale user-item interactions, which efficiently captures the correlations of Hawkes processes in different dimensions. Specifically, a Hawkes process is used to model the interaction sequence of each user-item pair. The parameter matrices for all processes, such as the base intensity matrix and self-exciting coefficient matrix, are assumed to behave as low-rank matrices in the neighborhoods of certain user-item combinations. Each parameter matrix is expressed as a smoothed aggregation of several low-rank matrices that approximate the parameters in local neighborhoods. We adopt non-parametric kernel smoothing to aggregate several local models into a unified model approximation. Based on the local low-rank approximation, the Hawkes processes for all user-item pairs are correlated due to the similarities between local mappings of the parameter matrices.

2.2 Model

We first introduce the background of Hawkes processes in Section 2.2.1 and then present the local low-rank Hawkes processes in Section 2.2.2 and 2.2.3. We list key notation in Table 2.1.

2.2.1 Background on Hawkes Process

A temporal point process is a random process [17, 1] and the realization of the process consists of a list of discrete temporal events $\mathcal{T} = \{t_i\}_{i=1}^n$. It is basically a counting process that counts the cumulative number of events $\{N(t), t \geq 0\}$ occurring right before time t. A counting process is also a submartingale, *i.e.*, $\mathbb{E}[N(t)|\mathcal{T}_{t'}] \geq N(t')$ for all $t > t'$, where $\mathcal{T}_{t'} = \{t_i | t_i < t'\}_{i=1}^n$

Table 2.1: Key notation.

Variable	Description
m	number of users
n	number of items
$\mathcal{T}^{u,i}$	a list of discrete temporal events for user-item pair (u, i)
$\mathcal{O}$	observed sequences of all user-item pairs
P	number of pairs that contain a sequence of observed events in $\mathcal{O}$
$t_i^{u,i}$	i-th event in $\mathcal{T}^{u,i}$
$\lambda(t)$	Hawkes process intensity function
η	base intensity in Hawkes process
α	self-exciting coefficient in Hawkes process
$\lambda_{(u,i)}(t)$	Hawkes process intensity for user-item pair (u, i)
$\kappa_\sigma(t)$	kernel function in Hawkes process with bandwidth σ
$\boldsymbol{H}_{u,i}$	(u, i)-th entry of the base intensity matrix for user-item pair (u, i)
$\boldsymbol{A}_{u,i}$	(u, i)-th entry of the self-exciting coefficient matrix for user-item pair (u, i)
q	number of anchor points
s	user-item pair (u, i)
τ	τ-th entry of a list of q anchor points
s_τ	user-item anchor pair (a_τ, b_τ) for the τ-th anchor points
$\boldsymbol{H}^{s_\tau}$	base intensity matrix for the τ-th anchor point pair s_τ
$\boldsymbol{A}^{s_\tau}$	self-exciting coefficient matrix for the τ-th anchor point pair s_τ
$K_h(s_1, s_2)$	smoothing kernel for user-item pairs s_1 and s_2 with bandwidth h
$K_h^{s_\tau}$	matrix for τ-th anchor point whose (u, i)-th entry is $K_h(s_\tau, (u, i))$
$\boldsymbol{H}'$, $\boldsymbol{K}'$, $\boldsymbol{A}'$	block matrices defined in eq. (2.8)
$\boldsymbol{M}\{u, i\}$	vector defined in eq. (2.9), where $\boldsymbol{M}$ can be $\boldsymbol{H}'$, $\boldsymbol{K}'$ and $\boldsymbol{A}'$
λ, β	model trade-off factors for constraints
$\boldsymbol{X}$	model parameter block matrix $[\boldsymbol{H}'; \boldsymbol{A}']$
$\boldsymbol{Z}$	auxiliary variable block matrix $[\boldsymbol{Z}_1; \boldsymbol{Z}_2]$
ρ	regularization factor
$\boldsymbol{X}^{s_\tau}$	τ-th local model parameter block matrix $[\boldsymbol{H}^{s_\tau}; \boldsymbol{A}^{s_\tau}]$
λ^τ, β^τ	τ-th local model trade-off factors for constraints
ρ^τ	τ-th local model regularization factor

denotes the history up to but not including time t'. A temporal point process can be characterized by the conditional intensity function $\lambda(t)$, which models the occurrence of the next event given all the previous events.

The functional form of the intensity function characterizes the temporal point process. For example, the intensity of a homogeneous Poisson process is constant over time, *i.e.*, $\lambda(t) = \eta \geq 0$. Alternatively, the Hawkes process, a conditional Poisson process, is particularly useful for modeling the mutual excitation between events. For example, the intensity can be defined as:

$$\lambda(t) = \eta + \alpha \sum_{t_i \in \mathcal{T}_t} \kappa_\sigma(t - t_i), \tag{2.1}$$

where $\kappa_\sigma(t) := \exp(-t/\sigma)$ is an exponential kernel function capturing temporal dependencies, $\eta \geq 0$ is a base intensity capturing the long-term incentive to generate events, and $\alpha \geq 0$ is the coefficient that magnifies the influence of each previous event.

Given a collection of events between m users and n items, the occurrences of user u's interaction events with item i can be modeled as a self-exciting Hawkes process [35], *i.e.*:

$$\lambda_{(u,i)}(t) = \boldsymbol{H}_{u,i} + \boldsymbol{A}_{u,i} \sum_{t_j^{u,i} \in \mathcal{T}_t^{u,i}} \kappa_\sigma(t - t_j^{u,i}), \tag{2.2}$$

where $\boldsymbol{H}$ denotes an $m \times n$ matrix with the (u, i)-th entry equal to the non-negative base intensity for user-item pair (u, i), and $\boldsymbol{A}$ denotes an $m \times n$ matrix with the (u, i)-th entry equal to the self-exciting coefficient for user-item pair (u, i). 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 induced between user u and item i up to but not including time t. In traditional approaches [86, 20], the two parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ are assumed to have low-rank structures.

A univariate Hawkes process can be extended to a multivariate Hawkes process [35, 56] to handle a multi-dimensional (e.g., $m \times n$) random process where different dimensions are correlated with each other. However, in most real-world events involving large dimensions m

and n, the parameter estimation of an MHP becomes inefficient [22, 45, 32].

2.2.2 Local Low-Rank Hawkes Process

Assuming that the mapping from user-item pairs to parameters is slowly varying, the parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ for all user-item pairs $s = (u, i) \in [m] \times [n]$ can be characterized by a smoothed combination of multiple low-rank matrices in a way similar to [51]. Specifically, we assume that there exists a metric over the user-item space $[m] \times [n]$. The distance between pair $s_1 = (a_1, b_1)$ and pair $s_2 = (a_2, b_2)$ is denoted by $d(s_1, s_2) = d((a_1, b_1), (a_2, b_2))$, which reflects the similarity between rows a_1 and a_2 and columns b_1 and b_2. We assume that there is a set of $q < m \cdot n$ anchor user-item pairs and each of them is associated with a base intensity matrix $\boldsymbol{H}^{s_\tau}$ and a self-exciting coefficient matrix $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, ..., q$. If $d(s_1, s_2)$ is small, $\boldsymbol{H}^{s_1}$ and $\boldsymbol{A}^{s_1}$ are similar to $\boldsymbol{H}^{s_2}$ and $\boldsymbol{A}^{s_2}$, respectively, by their spatial proximity in the embedding $\mathbb{R}^{m \times n}$. Typically, for an anchor pair $s_\tau = (a_\tau, b_\tau) \in [m] \times [n]$, the neighborhood $\{s' : d(s_\tau, s') < h\}$ in the original matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ can be approximated by the corresponding entries of matrices $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$.

Furthermore, we recover the mapping parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ from aggregating a set of $q < m \cdot n$ matrices without imposing a specific function form. Following common non-parametric approaches, we define a smoothing kernel $K_h(s_1, s_2)$, $s_1, s_2 \in [m] \times [n]$ for user-item pairs, which is a non-negative symmetric unimodal function parameterized by a bandwidth parameter $h > 0$. There are many popular choices of smoothing kernels, such as the Gaussian Kernel, Logistic Kernel, Sigmoid Kernel, and Silverman Kernel, defined as follows, respectively:

$$K_h(s_1, s_2) \propto \exp(-\frac{1}{2} h^{-2} d(s_1, s_2)^2), \tag{2.3}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + 2 + \exp(-d(s_1, s_2)/h)}, \tag{2.4}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + \exp(-d(s_1, s_2)/h)}, \tag{2.5}$$

$$K_h(s_1, s_2) \propto \exp(-\frac{|d(s_1, s_2)/h|}{\sqrt{2}}) \cdot \sin(\frac{|d(s_1, s_2)/h|}{\sqrt{2}} + \frac{\pi}{4}). \tag{2.6}$$

We adopt a type of locally constant kernel regression [82] to aggregate multiple local matrices. For simplicity, we use the same smoothing kernel and the same bandwidth for base intensity η and coefficient α. That is, for each user-item pair $s = (u, i)$, the occurrences of user u's interactions with item i are modeled as a local low-rank Hawkes process with the following intensity:

$$\lambda_s(t) = \sum_{\tau=1}^{q} \frac{K_h(s_\tau, s)}{\sum_{k=1}^{q} K_h(s_k, s)} [\boldsymbol{H}_s^{s_\tau} + \boldsymbol{A}_s^{s_\tau} \sum_{t_j^s \in \mathcal{T}_t^s} \kappa_\sigma(t - t_j^s)], \tag{2.7}$$

where $\boldsymbol{H}_s^{s_\tau}$ and $\boldsymbol{A}_s^{s_\tau}$ are the s-th entry of the τ-th base intensity matrix $\boldsymbol{H}^{s_\tau}$ and self-exciting matrix $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, ..., q$, respectively. Note that we have matrix index $s = (u, i)$. Since the users and the items in each matrix can be grouped into a limited number of sets with similar types, we assume that $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$ have low-rank structures. This means that the nuclear norms of the parameter matrices, $\|\boldsymbol{H}^{s_\tau}\|_*$ and $\|\boldsymbol{A}^{s_\tau}\|_*$, are small. Therefore, the mapping parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ in eq. (2.2) have local low-rank structures, and the local low-rank Hawkes process in eq. (2.7) is actually based on the weighted summation of q low-rank Hawkes processes in eq. (2.2). Specifically, our local low-rank Hawkes model is equivalent to the low-rank Hawkes model [20] when the number of anchor points is equal to one, i.e., $q = 1$.

To simplify the notation, we denote by $K_h^{(a,b)}$ the matrix whose (i, j)-entry is $K_h((a, b), (i, j))$. Given a series of anchor points, e.g., $s_\tau \in 1, ..., q$, let $C_s = \sum_{k=1}^{q} K_h(s_k, s)$, the denominator of which is the summation of the kernel weights and is actually a constant for each (u, i) pair. We further create three block matrices $\boldsymbol{H}'$, $\boldsymbol{K}'$, and $\boldsymbol{A}' \in \mathbb{R}^{m \times (q*n)}$ by concatenating a set of matrices

$\boldsymbol{H}^{s_\tau}$, $K_h^{s_\tau}$, and $\boldsymbol{A}^{s_\tau}$ as follows:

$$\boldsymbol{H}' = [\boldsymbol{H}^{s_1}, ..., \boldsymbol{H}^{s_q}], \boldsymbol{A}' = [\boldsymbol{A}^{s_1}, ..., \boldsymbol{A}^{s_q}], \boldsymbol{K}' = [K_h^{s_1}, ..., K_h^{s_q}]. \tag{2.8}$$

Also, let $\boldsymbol{M}\{u, i\}$ be a vector extracted from a matrix $\boldsymbol{M}$ for each (u, i) pair:

$$\boldsymbol{M}\{u, i\} = [\boldsymbol{M}^{s_1}(u, i), ..., \boldsymbol{M}^{s_q}(u, i)], \tag{2.9}$$

where $\boldsymbol{M}$ can be any of the three matrices $\boldsymbol{H}'$, $\boldsymbol{A}'$, and $\boldsymbol{K}'$.

2.2.3 Objective Function

Based on the survival analysis theory [1], the likelihood of observing a sequence of events $\mathcal{T} = \{t_i\}_{i=1}^n$ is $\prod_{t_i \in \mathcal{T}} \lambda(t_i) \cdot \exp(-\int_0^T \lambda(\tau) d(\tau))$, where T is the total observation time. Specifically, let $\mathcal{T}^{u,i}$ be the set of interaction events between entities u and i. The log-likelihood of observing each sequence $\mathcal{T}^{u,i}$ is:

$$\mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) = \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}_{u,i}^\top \Phi_j^{u,i}) - \boldsymbol{X}_{u,i}^\top \Psi^{u,i}, \tag{2.10}$$

where:

$$\begin{aligned} \boldsymbol{X}_{u,i} =& (\boldsymbol{H}'\{u, i\}, \boldsymbol{A}'\{u, i\})^\top, \\ \Phi_j^{u,i} =& C_{u,i}^{-1}(\boldsymbol{K}'\{u, i\} \cdot 1, \ \ \boldsymbol{K}'\{u, i\} \cdot \sum_{t_k^{u,i} < t_j^{u,i}} \kappa_\sigma(t_j^{u,i} - t_k^{u,i}))^\top, \\ \Psi^{u,i} =& C_{u,i}^{-1}(\boldsymbol{K}'\{u, i\} \cdot T, \ \ \boldsymbol{K}'\{u, i\} \cdot \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^T \kappa_\sigma(t - t_j^{u,i}) dt)^\top. \end{aligned} \tag{2.11}$$

As a result, the log-likelihood of observing all user-item interaction sequences $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$ is a summation of terms by $\mathcal{L}(\mathcal{O}) = \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i})$. We can obtain the model parameters $\boldsymbol{X}$ by minimizing the following objective function:

$$OPT = \min_{\boldsymbol{X}} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) + h(\boldsymbol{X}), \qquad s.t.\ \boldsymbol{X} \geq \mathbf{0}, \tag{2.12}$$

where $h(\boldsymbol{X}) = \lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_*$, $\boldsymbol{X} = [\boldsymbol{H}'; \boldsymbol{A}']$, and λ and β control the trade-off between the constrains. The nuclear norm $\|\cdot\|_*$ is a summation of all singular values and it can be used as a convex surrogate for the matrix rank [73]. Thus, minimizing $\|\boldsymbol{H}'\|_*$ and $\|\boldsymbol{A}'\|_*$ ensures each $\boldsymbol{H}^{s_\mathcal{T}}$ and $\boldsymbol{A}^{s_\mathcal{T}}$ to be low-rank. After obtaining $\boldsymbol{X}$, we can use eq. (2.7) to compute the intensity.

2.3 Parameter Estimation and the Algorithms

To estimate model parameters, we introduce an efficient framework to optimize the objective in eq. (2.12). Specifically, we introduce the latest Primal Averaging Conditional Gradient (PA-CndG) algorithm [50] based on the Proximal Gradient (PG) method [49, 66]. The algorithm, also referred to as the global approach, is described in Section 2.3.1. The convergence analysis of the algorithm is described in Section 2.3.2. We further present a parallel algorithm to increase the computation efficiency in Section 2.3.3. Finally, the computational complexity analysis is described in Section 2.3.4.

2.3.1 Approximate Function and Gradient Update

Directly solving the objective in eq. (2.12) is difficult because the non-negative constraints are coupled together with the non-smooth nuclear norm. To tackle the difficulties, we approximate eq. (2.12) by adopting a penalty method [20, 84]. Given $\rho > 0$, we introduce an auxiliary variable $\boldsymbol{Z} = [\boldsymbol{Z}_1; \boldsymbol{Z}_2]$ with the squared Frobenius norm, which leads to the new formulation in eq. (2.13):

$$\widehat{OPT} = \min_{\boldsymbol{X},\boldsymbol{Z}} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) + h(\boldsymbol{Z}) + g(\boldsymbol{X}, \boldsymbol{Z}), \qquad s.t.\ \boldsymbol{X} \geq \mathbf{0}, \tag{2.13}$$

where $g(\boldsymbol{X}, \boldsymbol{Z}) = \rho \|\boldsymbol{H}' - \boldsymbol{Z}_1\|_F^2 + \rho \|\boldsymbol{A}' - \boldsymbol{Z}_2\|_F^2$. In this formulation of eq. (2.13), the nuclear norm regularization terms and the non-negativity constraints are handled separately. The approximate objective can always be the upper bound of the real objective given the bounded ρ [20].

For simplicity, we set:

$$f(\boldsymbol{X}, \boldsymbol{Z}) = -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) + g(\boldsymbol{X}, \boldsymbol{Z}), \tag{2.14}$$

and the objective function becomes:

$$\widehat{OPT} = F(\boldsymbol{X}, \boldsymbol{Z}) = f(\boldsymbol{X}, \boldsymbol{Z}) + h(\boldsymbol{Z}), \qquad s.t.\ \boldsymbol{X} \geq \boldsymbol{0}, \tag{2.15}$$

Note that $f(\cdot)$ is convex and Lipschitz continuous gradient (L-smooth), and $h(\cdot)$ is convex.

Algorithm 1: Local Low-Rank Hawkes

```
Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate $\xi_k$; parameters $\rho$, $\lambda$, $\beta$;
       number of anchor points $q$; kernel function $K(\cdot)$ of widths $h_1$, $h_2$; step size
       $\gamma_k \in [0, 1]$;
Output: $\boldsymbol{X} = [\boldsymbol{H}'; \boldsymbol{A}']$, which is the block matrix
for $\tau = 1 \to q$ do
    $(a_\tau, b_\tau) :=$ a random selected $(u, i)$ pair;
    for $i = 1 \to m$ do
        $K_{h_1}^{a_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(a_\tau, i)^2)$;
    end
    for $j = 1 \to n$ do
        $K_{h_2}^{b_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(b_\tau, j)^2)$;
    end
end
Choose to initialize $\boldsymbol{U}_1^0$;
Set $\boldsymbol{X}^0 = \boldsymbol{Z}^0 = \boldsymbol{U}_1^0 = \boldsymbol{U}_2^0$;
for $k \leftarrow$ 1 to MaxIter do
    Set $\boldsymbol{Y}_1^{k-1} = (1 - \gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^{k-1}$;
    Set $\boldsymbol{Y}_2^{k-1} = (1 - \gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^{k-1}$;
    Compute the proximal operator for $\boldsymbol{X}$:
    $\boldsymbol{U}_1^k = (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1(f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})))_+$;
    Use a local linear expansion of $f$ for $\boldsymbol{Z}$:
    $\boldsymbol{U}_2^k = arg\min_{\boldsymbol{Z}}\{\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z} \rangle + h(\boldsymbol{Z})\}$;
    Set $\boldsymbol{X}^k = (1 - \gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^k$;
    Set $\boldsymbol{Z}^k = (1 - \gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^k$;
end
```

As shown in Algorithm 1, we apply gradient update for model parameters X and Z in each iteration and keep three interdependent sequences U^k, X^k, and Y^k based on the schema in [66].

Specifically, we directly compute the proximal operator for $\boldsymbol{X}$ with the constraint in Algorithm 1 as:

$$\begin{aligned}\boldsymbol{U}_1^k &= \arg\min_{\boldsymbol{U}_1^k \geq 0} \{\frac{1}{2\xi_k} \|\boldsymbol{U}_1^k - (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}))\|^2\} \\ &= (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 (f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})))_+. \end{aligned} \tag{2.16}$$

Note that $h(\boldsymbol{Z})$ only has variable $\boldsymbol{Z}$, so $h(\cdot) = 0$, which means that it is just normal Projected Gradient Descent (PGD). Besides, $(\cdot)_+$ in Algorithm 1 sets the negative coordinates to zero.

For $\boldsymbol{Z}$, we do not directly calculate using eq. (2.16). Instead, we use a local linear expansion to approximate it, which is known as conditional gradient. Specifically, it differs from traditional conditional gradient method in the way that the search direction $\nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})$ is defined. It can be viewed as a variant of Nesterov's method [66] and is obtained by replacing the prox-mapping with a simpler linear expansion:

$$\boldsymbol{U}_2^k = arg \min_{\boldsymbol{Z}} \{\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z}\rangle + h(\boldsymbol{Z}). \tag{2.17}$$

Specifically, this part can be solved by first calculating the top singular vector pairs of $-\nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})$ and then using a line search to produce a scaling factor [93, 20].

2.3.2 Convergence Analysis

For PGD method, the algorithm achieves the well-known optimal rate $O(1/k)$, i.e., a rate of $O(1/\epsilon)$ given learning rate $\xi_k \leq 1/L$, and for PA-CndG method, it also reaches $O(1/k)$ given the step size policy (1): $\gamma_k = \frac{2}{k+1}$ or (2): $\gamma_k = \arg\min_{\gamma \in [0,1]} f((1-\gamma)\boldsymbol{X}^{k-1} + \gamma \boldsymbol{U}_1^k, (1-\gamma)\boldsymbol{Z}^{k-1} + \gamma \boldsymbol{U}_2^k)$ [50]. Generally, the algorithm should still reach the optimal rate $O(1/k)$ by properly choosing the step size parameter and the learning rate. We have the convergence results for Algorithm 1 in theorem 1, followed by the proof.

Theorem 1. *Let $\{\boldsymbol{Z}^k\}$, $\{\boldsymbol{X}^k\}$, $\{\boldsymbol{U}_1^k\}$, and $\{\boldsymbol{U}_2^k\}$ be the sequences generated by Algorithm 1 with* *step size $\gamma_k = \frac{2}{k+1}$ and learning rate $\xi_k \leq 1/L$. Then we have:*

$$F(\boldsymbol{X}^k, \boldsymbol{Z}^k) - F^* \leq \frac{5LD_{max}^2}{k+1}, \tag{2.18}$$

where L is the Lipschitz constant of $\nabla f(x, z)$.

Proof. Define:

$$l_f(x, z; y_1, y_2) = f(x, z) + \langle \nabla_1 f(x, z), y_1 - x \rangle + \langle \nabla_2 f(x, z), y_2 - z \rangle. \tag{2.19}$$

For $\boldsymbol{X}, \boldsymbol{Z} \in \Omega$, f is Lipschitz continuous gradient and:

$$f(y_1, y_2) \leq l_f(x, z; y_1, y_2) + \frac{L}{2}\|y_1 - x\|^2 + \frac{L}{2}\|y_2 - z\|^2. \tag{2.20}$$

First note that:

$$\begin{aligned} \boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1} &= \gamma_k(\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}), \\ \boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1} &= \gamma_k(\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}). \end{aligned} \tag{2.21}$$

Hence, using the definitions of $\boldsymbol{X}^k$ and $\boldsymbol{Z}^k$ in Algorithm 1, we have:

$$\begin{aligned} f(\boldsymbol{X}^k, \boldsymbol{Z}^k) &\leq l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^k, \boldsymbol{Z}^k) + \frac{L}{2}\|\boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1}\|^2 + \frac{L}{2}\|\boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1}\|^2 \\ &= (1-\gamma_k) l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) + \gamma_k l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{U}_1^k, \boldsymbol{U}_2^k) \\ &+ \frac{L}{2}\gamma_k^2\|\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}\|^2 + \frac{L}{2}\gamma_k^2\|\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}\|^2. \end{aligned} \tag{2.22}$$

For simplicity, define the Bregman divergence $D(x, x') = \|x - x'\|^2$. From eq. (2.16), we know it is actually PGD method with $f(\cdot)$ as Lipschitz continuous gradient and constrained to convex set Ω. Based on the definition of the convex hull and the property of PGD, we have the following property:

$$\langle \boldsymbol{U}_1 - \boldsymbol{Y}_1^k, (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})) - \boldsymbol{Y}_1^k \rangle \leq 0, \qquad \forall\, \boldsymbol{U}_1 \in \Omega. \tag{2.23}$$

Using eq. (2.23) and the definition of $\boldsymbol{U}_2^k$ in Algorithm 1, we have:

$$\langle \nabla_1 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Y}_1^k - \boldsymbol{U}_1^* \rangle \leq -\frac{1}{2\xi_k} \boldsymbol{D}(\boldsymbol{Y}_1^k, \boldsymbol{Y}_1^{k-1}) + \frac{1}{2\xi_k} [\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] \tag{2.24}$$

and

$$\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{U}_2^k \rangle + h(\boldsymbol{U}_2^k) \leq \langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{U}_2^* \rangle + h(\boldsymbol{U}_2^*). \tag{2.25}$$

Then noting that $D(x, x') \geq 0$ and using the convexity of $f(\cdot)$ and $h(\cdot)$ together with the definition of $\boldsymbol{Z}^k$ in Algorithm 1 and eqs. (2.22), (2.24) and (2.25), we end up with:

$$\begin{aligned}
F(\boldsymbol{X}^k, \boldsymbol{Z}^k) \leq{}& (1-\gamma_k) f(\boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) + \gamma_k l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{U}_1^*, \boldsymbol{U}_2^*) \\
&+ \frac{\gamma_k}{2\xi_k} [\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] + \gamma_k (h(\boldsymbol{U}_2^*) - h(\boldsymbol{U}_2^k)) + h(\boldsymbol{Z}^k) \\
&+ \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k, \boldsymbol{U}_1^{k-1}) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k, \boldsymbol{U}_2^{k-1}) \\
\leq{}& (1-\gamma_k) f(\boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) + \gamma_k f(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) + \frac{L}{2} \gamma_k [\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] \\
&+ \gamma_k (h(\boldsymbol{U}_2^*) - h(\boldsymbol{U}_2^k)) + h(\boldsymbol{Z}^k) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k, \boldsymbol{U}_1^{k-1}) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k, \boldsymbol{U}_2^{k-1}) \\
\leq{}& (1-\gamma_k) F(\boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) + \gamma_k F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) + \frac{L}{2} \gamma_k [\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] \\
&- \gamma_k h(\boldsymbol{U}_2^k) + h(\boldsymbol{Z}^k) - (1-\gamma_k) h(\boldsymbol{Z}^{k-1}) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k, \boldsymbol{U}_1^{k-1}) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k, \boldsymbol{U}_2^{k-1}) \\
\leq{}& (1-\gamma_k) F(\boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) + \gamma_k F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) + \frac{L}{2} \gamma_k [\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] \\
&+ \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k, \boldsymbol{U}_1^{k-1}) + \frac{L}{2} \gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k, \boldsymbol{U}_2^{k-1}).
\end{aligned} \tag{2.26}$$

Subtracting $F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*)$ from both sides of the above inequality, we have:

$$\begin{aligned} F(\boldsymbol{X}^k, \boldsymbol{Z}^k) - F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) &\le (1-\gamma_k)(F(\boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) - F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*)) \\ &+ \frac{L}{2}\gamma_k[\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k, \boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k, \boldsymbol{U}_2^{k-1}). \end{aligned} \tag{2.27}$$

In view of Lemma 1 of [50] and the definition of γ_k and Γ_k, it is easy to verify that $\frac{\gamma_k^2}{\Gamma_k} = \frac{2k}{k+1} \le 2$ and $\frac{\gamma_i}{\Gamma_i} = i \le k$, which implies that:

$$\begin{aligned} F(\boldsymbol{X}^k, \boldsymbol{Z}^k) - F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) &\le \Gamma_k(1-\gamma_1)(F(\boldsymbol{X}^0, \boldsymbol{Z}^0) - F(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*)) \\ &+ \frac{\Gamma_k L}{2}\sum_{i=1}^{k}\{\frac{\gamma_i}{\Gamma_i}[\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{i-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^i)] + \frac{\gamma_i^2}{\Gamma_i}[\boldsymbol{D}(\boldsymbol{U}_1^i, \boldsymbol{U}_1^{i-1}) + \boldsymbol{D}(\boldsymbol{U}_2^i, \boldsymbol{U}_2^{i-1})]\}. \end{aligned} \tag{2.28}$$

Let $D_{max} = \max_{x,y\in\Omega}\|x-y\|$ and note that $D(x,x') \ge 0$. We finally have:

$$F(\boldsymbol{X}^k, \boldsymbol{Z}^k) - F^* \le \frac{L}{k(k+1)}\{k\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^0) + 2\sum_{i=1}^{k}[\boldsymbol{D}(\boldsymbol{U}_1^i, \boldsymbol{U}_1^{i-1}) + \boldsymbol{D}(\boldsymbol{U}_2^i, \boldsymbol{U}_2^{i-1})]\} \le \frac{5LD_{max}^2}{k+1}. \tag{2.29}$$

Therefore, the algorithm still achieves the optimal rate $O(1/k)$, i.e., a rate of $O(1/\epsilon)$. □

2.3.3 Parallel Algorithm

As mentioned earlier, the above global algorithm may be computational expensive when the number of anchor points q increases to an extremely large value. We further speed up the algorithm to accommodate the need of a large number of anchor points q to fit big industry data. To this end, we first rewrite the optimal function in the form of eq. (2.32). We show in theorem 2 that when λ^τ and β^τ are properly chosen, the two formulations will result in the same optimum. As all the variables $\{\boldsymbol{X}^{s_\tau} = [\boldsymbol{H}^{s_\tau}; \boldsymbol{A}^{s_\tau}]\}_{\tau=1}^q$ are independent, we develop the parallel method in Algorithm 2 that optimizes each block matrix $\{\boldsymbol{X}^{s_\tau}\}_{\tau=1}^q$ separately. Hence, it allows us to deal with the objective function in parallel and makes the algorithm more efficient for big data.

Denote the log-likelihood of observing sequence $\mathcal{T}^{u,i}$ mapping to a specific anchor point $s_\tau = (a_\tau, b_\tau)$ as:

$$\mathcal{L}^{s_\tau}(\mathcal{T}^{u,i} \mid \boldsymbol{X}^{s_\tau}) = \frac{1}{q}\{\sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}) - \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i}\}, \tag{2.30}$$

where:

$$\begin{aligned}
\boldsymbol{X}(s_\tau)_{u,i} =& (\boldsymbol{H}^{s_\tau}(u,i), \boldsymbol{A}^{s_\tau}(u,i))^\top, \\
\Phi(s_\tau)_j^{u,i} =& q(1, \sum_{t_k^{u,i} < t_j^{u,i}} \kappa_\sigma(t_j^{u,i} - t_k^{u,i}))^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}, \\
\Psi(s_\tau)^{u,i} =& q(T, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{T} \kappa_\sigma(t - t_j^{u,i})dt)^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}.
\end{aligned} \tag{2.31}$$

Then we define the parallel objective function as:

$$OPT_p = \min_{\boldsymbol{X}^{s_\tau}, \boldsymbol{Z}^{s_\tau}} \frac{1}{|\mathcal{O}|} \sum_{\tau=1}^{q} \{\sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}^{s_\tau}(\mathcal{T}^{u,i} | \boldsymbol{X}^{s_\tau}) + h_{s_\tau}(\boldsymbol{Z}^{s_\tau})\}, \qquad s.t.\ \boldsymbol{X} \geq \boldsymbol{0}, \tag{2.32}$$

where $h_{s_\tau}(\boldsymbol{Z}^{s_\tau}) = \lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*$.

Theorem 2. *With the condition that λ^τ and β^τ for $\tau = 1, ..., q$ satisfy eq.* (2.33)*, the optimal value OPT_p in eq.* (2.32) *coincides with the global optimal value OPT in eq.* (2.15)*.*

$$\lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_* \leq \sum_{\tau=1}^{q} (\lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*). \tag{2.33}$$

Proof. For a real convex function $\varphi(\cdot)$, a set of numbers $x_1, x_2, ..., x_n$, and positive weights α_i, Jensen's inequality can be stated as:

$$\varphi(\frac{\sum \alpha_i x_i}{\alpha_i}) \leq \frac{\sum \alpha_i \varphi(x_i)}{\sum \alpha_i}. \tag{2.34}$$

The equality holds if and only if $x_1 = x_2 = ... = x_n$ or $\varphi(\cdot)$ is linear. Specifically, eq. (2.34) becomes:

$$\varphi(\frac{\sum x_i}{n}) \leq \frac{\sum \varphi(x_i)}{n} \tag{2.35}$$

if the weighs α_i are equal.

As $-log(\cdot)$ is convex, we rewrite eq. (2.10) based on eq. (2.35) as:

$$\begin{aligned} &-\mathcal{L}(\mathcal{T}^{u,i} \mid \{\boldsymbol{X}^{s_\tau}\}_{\tau=1}^{q}) = -\sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}/q) + \sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i}/q \\ &\leq -\sum_{\tau=1}^{q} \{ \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}) - \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i} \}/q = -\sum_{\tau=1}^{q} \mathcal{L}^{s_\tau}(\mathcal{T}^{u,i} \mid \boldsymbol{X}^{s_\tau}). \end{aligned} \tag{2.36}$$

Given eq. (2.33), we have:

$$h(\boldsymbol{Z}) = \lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_* \leq \sum_{\tau=1}^{q} (\lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*) = \sum_{\tau=1}^{q} h_{s_\tau}(\boldsymbol{Z}^{s_\tau}). \tag{2.37}$$

Therefore, plugging eqs. (2.36) and (2.37) into the previous objective function in eq. (2.12), we have $OPT \leq OPT_p$ and readily arrive at the theorem. □

Therefore, we can optimize the parallel objective function in eq. (2.32) separately by using the parallel algorithm to approximate the parameter estimation. As the form of the objective function is the same as the global one, we can still use the global updating approach. The details are described in Algorithm 2.

2.3.4 Computational Complexity

Given a collection of interaction events between m users and n items, we assume for the worst case each user-item pair has a sequence of events observed. The time complexity of calculating the gradient of each user-item entry of each parameter matrix is a constant C, and thus the total time complexity of the global algorithm with q anchor points is $O(N^2 qC/\epsilon)$, where $N = max\{m, n\}$, since we have $2 \times m \times n \times q$ entries in the global model parameter matrix $\boldsymbol{X}$

```
Algorithm 2: Local Low-Rank Hawkes Parallel
Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate $\xi_k$; parameters $\rho$, $\lambda$, $\beta$;
       number of anchor points $q$; kernel function $K(\cdot)$ of widths $h_1$, $h_2$; step size
       $\gamma_k \in [0,1]$;
Output: $\{\boldsymbol{X}^{s_\tau} = [\boldsymbol{H}^{s_\tau}; \boldsymbol{A}^{s_\tau}]\}_{\tau=1}^{q}$, which are the set of local parameter matrices:
for $\tau = 1, ..., q$ in parallel do
    $(a_\tau, b_\tau) :=$ a random selected $(u,i)$ pair;
    for $i = 1 \to m$ do
        $K_{h_1}^{a_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(a_\tau, i)^2)$;
    end
    for $j = 1 \to n$ do
        $K_{h_2}^{b_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(b_\tau, j)^2)$;
    end
    Choose to initialize $\boldsymbol{U}_1^0$;
    Set $\boldsymbol{X}^0 = \boldsymbol{Z}^0 = \boldsymbol{U}_1^0 = \boldsymbol{U}_2^0$;
    for $k \leftarrow 1$ to MaxIter do
        Set $\boldsymbol{Y}_1^{k-1} = (1-\gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^{k-1}$;
        Set $\boldsymbol{Y}_2^{k-1} = (1-\gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^{k-1}$;
        Compute the proximal operator for $\boldsymbol{X}$:
        $\boldsymbol{U}_1^k = (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 (f_{s_\tau}(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})))_+$;
        Use a local linear expansion of $f$ for $\boldsymbol{Z}$:
        $\boldsymbol{U}_2^k = arg\min_{\boldsymbol{Z}}\{\langle \nabla_2 f_{s_\tau}(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z}\rangle + h_{s_\tau}(\boldsymbol{Z})\}$;
        Set $\boldsymbol{X}^k = (1-\gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^k$;
        Set $\boldsymbol{Z}^k = (1-\gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^k$;
    end
end
```

and the algorithm takes $O(1/\epsilon)$ iterations. In practice, we only need to compute the sequences of observed pairs that satisfy $\mathcal{T}^{u,i} \in \mathcal{O}$, and usually the entries are quite sparse in the real world dataset. For example, the ratio of the number of the observed user-item pairs to that of the total pairs ranges from 0.001 to 0.01 as shown in Table 2.2. Assume that there are $P \ll m \cdot n$ pairs with sequences of observed events in the dataset, the complexity of the global algorithm becomes $O(PqC/\epsilon)$. In summary, the complexity of the global algorithm increases as the size of the dataset or the number of anchor points increases. The parallel algorithm should be q times faster without the consideration of the communication cost. Specifically, by assuming q machines for computing, the algorithm can run in parallel to estimate the q local model parameters. Fewer entries in the local parameter block matrix $\boldsymbol{X}^{s_\tau}$ need to be computed in comparison with the case of matrix $\boldsymbol{X}$ in the global algorithm. In the end, the local parameters will be combined to obtain the final results. The parallel algorithm can speed up the global algorithm and achieves the complexity of $O(PC/\epsilon)$.

2.4 Distance Measure, Kernel Calculation, and Anchor Point Selection

The distance metric d such as eq. (2.3) that measures the similarities between users or items may be defined based on auxiliary content features such as user demographics and item genres. If no such information is available, the user and item similarities can be measured through partially observed user-item interaction matrix X, where each entry indicates the frequency of user-item interactions during a time period. Specifically, the distances between row vectors (for users) and between column vectors (for items) can be computed based on standard distance measures such as "Cosine" metric and "Arc-cosine" metric. The "Cosine" metric between a pair of vectors is defined as one minus the cosine of the angle between the vectors and the "Arc-cosine" metric is the inverse cosine function between vectors. When the matrix X is very sparse, we follow conventions as reported in [51] to factorize the matrix using standard incomplete SVD [4] to obtain the latent matrices. We also investigate different combination methods in Section 2.5.3 to construct a unified feature based on both user-item interactions and content features, and compare the performance.

In Sections 2.2.2 and 2.3, we assume a general kernel function denoted by $K_h(s_1, s_2)$, where $s_1, s_2 \in [m] \times [n]$. Similar to the related work [51], we assume a product form $K_h((a_1, b_1), (a_2, b_2)) = K_{h_1}(a_1, a_2) \cdot K'_{h_2}(b_1, b_2)$, where those two kernels are on the spaces $[m]$ and $[n]$, respectively. Due to the computation of the log function in eq. (2.10), we use the Gaussian kernel in eq. (2.3) for both K and K', as the kernel function is non-zero everywhere. The kernel function matrix for one anchor point can be expressed as $K_h^{s_\tau} = K_{h_1}^{a_\tau} \cdot K_{h_2}^{b_\tau} \in [m] \times [n]$, where $\tau = 1, ..., q$. Generally, the kernel bandwidth parameter represents the amount of smoothing and affects the estimation accuracy. We investigate the dependency of the model performance on kernel bandwidth in Section 2.5.3. We also compare the performance of kernel smoothing with different numbers of anchor points in Section 2.5.3.

There are several approaches of selecting the anchor points $s_1, ..., s_q$, which may affect the model performance. For simplicity, we randomly select anchor points from the observed (u, i) entries in the matrix in our algorithm. It is worth mentioning that the anchor points can be selected by other strategies such as pre-cluster processing, that is clustering the (u, i) pairs into q clusters and then selecting one anchor point from each cluster. There are several clustering methods such as K-Means Clustering and Hierarchical Clustering. For K-Means Clustering, the input features for each (u, i) pair could be the concatenated user and item latent features obtained by SVD. For Hierarchical Clustering, the input distance matrix is the one calculated by smoothing kernels. We carry out experiments to select anchor points from clusters via several cluster analysis methods in Section 2.5.3.

2.5 Experiments

In this section, we present the results of the experiments.

2.5.1 Datasets and Evaluation Criteria

We evaluate our model on the three real-world datasets and the details of each dataset are shown in Table 2.2.

IPTV dataset [87] records the viewing behaviors of 7100 users on 436 TV programs, e.g., what and when they watch, from January to November 2012. It contains 4726 (u, i) pairs with

Table 2.2: Dataset description.

Dataset	Users	Items	Events	Pairs	Item Features	Time Duration
IPTV	7100	436	$2.4M$	4726	1420	8040
Yelp	100	$17K$	$35K$	20246	823	44640
Reddit	1000	1403	$10K$	2053	35	4090

nearly 2.4M events and 1420 movie features such as genres. These features are only used for Coevolve baseline and experiments in Section 2.5.3 considering features with distance calculation. **Yelp**[1] is available from Yelp dataset challenge. We select users with at least 100 posts, and it contains 35k reviews for 17k businesses by 100 users in 11 years. **Reddit**[2] dataset contains a random selected 1000 users, 1403 groups, and 10k discussions events in January 2014.

We can evaluate the performance of our Hawkes model on three tasks:

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 $S_{(u,i)}(t) = \exp(-\int_{t_n^{u,i}}^t \lambda_{(u,i)}(\tau)d(\tau))$. According to the survival, we rank all the items in ascending order, and the ground-truth testing item should be at the top ideally. Following [84], we report *mean average rank* (MAR) of all testing cases. A smaller value of MAR indicates better predictive performance.

Time Prediction: Given a specific pair of user u and item i, we report the *mean absolute error* (MAE) [84, 20] of the next predicted time and the ground truth of testing time t. Specifically, we compute the predicted time by calculating the density as $f(t) = \lambda_{(u,i)}(t)S_{(u,i)}(t)$, and then use the expectation to predict the next event. Furthermore, we give the relative percentage of the prediction error (Err %).

Test Loss: It is defined as in the objective function eq. (2.13) with fixed coefficients of Hawkes processes learned using events in the training set.

2.5.2 Baseline Methods and Parameter Settings

Poisson process is a relaxation of Hawkes process with no triggering kernel capturing temporal dependencies. It only contains a base intensity η, which is a constant. The Poisson process

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

is a strong baseline in many cases, as most popular items usually have large base intensities.

PoissonTensor uses Poisson regression other than RMSE as the loss function to fit the number of events, which actually can be considered as the intensity in each discretized time slot [14]. Because the missing values are not random, simulating the values with Poisson distribution is more reasonable than with Gaussian. Once we get the values, 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.

LowRankHawkes is a Hawkes process based model [20] that can be seen as a relaxation of our model with only one anchor point. It assumes that all the (u, i) pairs are independent so there is no user-item interactions between pairs.

Coevolve is a coevolutionary latent feature process [84] which can be seen as a squared Hawkes process adding a base intensity. It uses user and item features as well as the interaction features between users and items, such as review features, to simulate the intensity of each (u, i) pair. In our experiments, we only use the item feature. If no features are provided, the model reduces to the Poisson process.

Parameter Settings: In the experiments, T is the length of the total time, and $p = 0.76$ is the proportion where we split the data. Specifically, we use the events before time $T \cdot p$ as the training data, and the rest of them as testing data. We do experiments on several types of kernels, and find that these do not affect the performance much. We use the Gaussian kernel with $h_1 = h_2 = 0.8$ and report the averaged results on the two tasks above.

2.5.3 Results

In Section 2.5.3, we first compare the results of our method with baseline methods using the global algorithm. We then analyze the model performance in terms of its dependency on feature integration methods, kernel bandwidth parameter, the number of anchor points, and anchor point selection approaches in Sections 2.5.3, 2.5.3, 2.5.3, and 2.5.3, respectively. Finally, we compare the model performance of the global and parallel algorithms in Section 2.5.3.

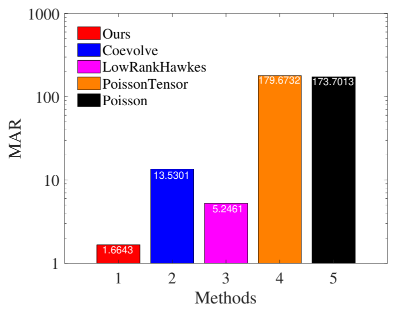
Comparison with Baselines

As described in Section 2.4, we follow [51] to factorize the user-item interaction matrix using standard incomplete SVD [4] to obtain the latent feature matrices. The kernel function and "Arc-cosine" metric measure the distances between different user-item pairs and a set of anchor points are random selected from the user-item dimension.

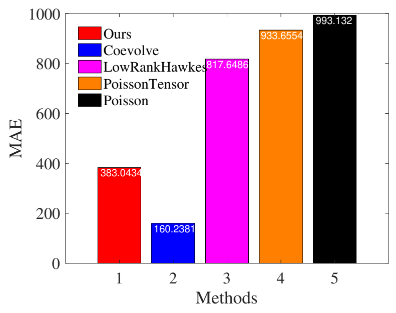
We show the results in Fig. 2.1 for IPTV, Yelp, and Reddit data, respectively. Generally, our model outperforms most other baseline methods in item prediction and returning time prediction. The main reason is that each (u, i) pair's intensity is simulated with its own sequence mapping to a total of q local models in our model. **Coevolve** relies on the auxiliary features. **LowRankHawkes** treats each (u, i) pair's process independently. **Poisson** and **PoissonTensor** simulate events without the history, and thus are lack of prediction power.

For IPTV and Reddit data, the exception occurs at the time prediction of **Coevolve**, because the auxiliary feature information is added to this model. The **Coevolve** method uses a weighted summation of all the events happened before the current event to simulate one (u, i) pair's intensity $\lambda^{u,i}(t)$. Therefore, the returning-time prediction is good since a large number of events are used to simulate the intensity function and a huge amount of auxiliary feature information is incorporated. However, the item rank prediction becomes worse [84] because the individual preferences are influenced by the general preference. Meanwhile, we can see that the Hawkes process based models, such as our model, **Coevolve**, and **LowRankHawkes**, get better performances when there are sufficient history events (with nearly 400 events per (u, i) pairs) in comparison with the Poisson related models.

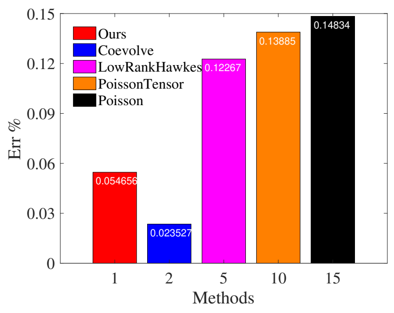
For Yelp data, as each (u, i) pair only has fewer than 3 events in average, the time prediction is similar among **LowRankHawkes** and **Poisson**, which means that the history is not such an important factor. In this time sparsity case, factorization model **PoissonTensor** gets better results than point process based models. Even adding some auxiliary features, the **Coevolve** model achieves comparable performance, our model performs the best without any features. As more information is used to simulate the Hawkes process for each (u, i) pair, our model integrates



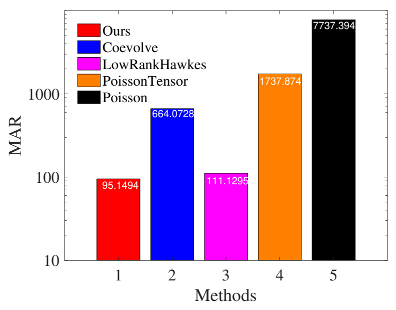
(a) Item relevance



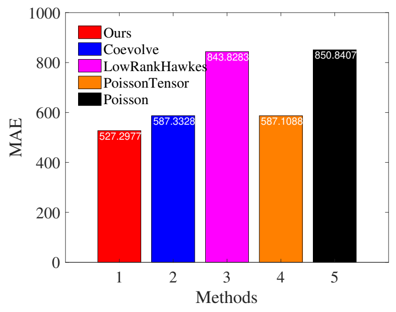
(b) Time prediction



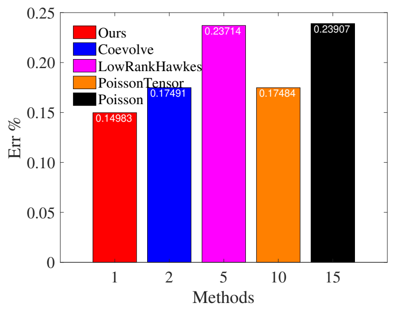
(c) Err %



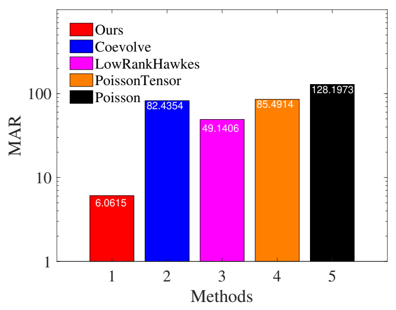
(d) Item relevance



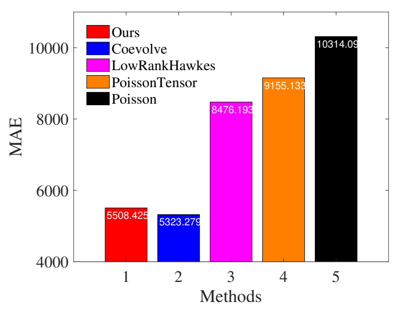
(e) Time prediction



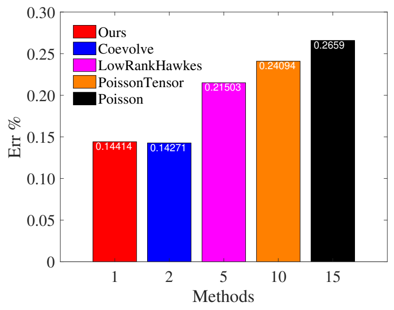
(f) Err %



(g) Item relevance



(h) Time prediction



(i) Err %

Figure 2.1: Prediction accuracy on IPTV (top row), Yelp (middle row), and Reddit (bottom row).

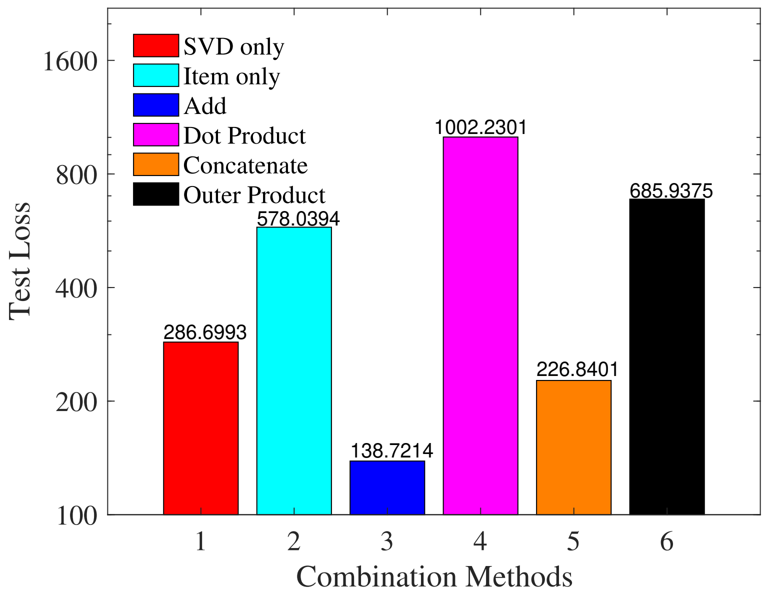


Figure 2.2: Test loss with respect to different combination methods of item features on IPTV dataset.

some interaction influences from similar groups. In other words, our model performs better on sequences without sufficient events.

Effect of Content Feature Integration

We investigate different combination methods using IPTV dataset to construct a unified item feature based on both user-item interactions and item genres, compute the distance between user and item similarities based on those features, and compare the model performance. We present six strategies: item collaborative features by factoring user-item interaction, which is the method we use previously (Collaborative); item content features (e.g., movie genres) (Content); addition of these two features; element-wise product of the two; concatenation of the two; and the outer product of the two flattened to one dimension. Since the content features are high dimensional and sparse, we first reduce the dimension of content features to the same dimension of item collaborative features. We then normalize these two types of features and finally integrate them to a unified feature vector.

Fig. 2.2 illustrates the performance of test loss with respect to different combination methods of item features on IPTV dataset. First of all, directly adopting items features through factoring user-item interactions is better than only adopting item content features. Second, it seems that addition and concatenation operations of these two features achieve better performance than only using one of them. However, element-wise product and outer product operations perform worse than using a single type of features. One of the reasons would be those two operations introduce redundant and noisy information. In addition, we adopt three strategies including Pricipal Component Analysis (PCA), Auto Encoder(AE), and Multi-dimensional Scaling (MDS) to reduce the dimensions of item content features. In our experiments, we find that PCA achieves the best performance, MDS is the second, and AE is the worst.

Effect of the Kernel Bandwidth

We investigate the effect of kernel smoothing on IPTV dataset. Generally, the bandwidth parameter represents the amount of smoothing with small values corresponding to narrow kernels and large values corresponding to wide kernels. As the bandwidth increases, the overlaps between these local models become more and more significant. In addition, the bias of each local model increases and the variance of each model decreases. The changes of test loss with respect to increasing smoothing kernel bandwidth are shown in Fig. 2.3. It is obvious that the best performance is achieved when the kernel bandwidth is in the range of $[h_1, h_2] \in [0.7, 0.8]$. The performance deteriorates as the kernel bandwidth deviates from an optimal bandwidth in the range of [0.7, 0.8]. Meanwhile, the performance is stable as long as the smoothing kernel bandwidth is selected in an appropriate range.

It is worth mentioning that we can calculate the smoothing kernel bandwidths for both users and items separately. The performance may be further enhanced by investigating different combinations of user kernel bandwidths and item kernel bandwidths. For simplicity, we use the same kernel bandwidth for users and items in our experiments.

Effect of the Number of Anchor Points

We also compare the performance of kernel smoothing with different numbers of anchor points in Table 2.3 for IPTV, Yelp, and Reddit datasets. The results are obtained using our global algorithm.

For IPTV and Reddit data, both item prediction and returning-time prediction are improved

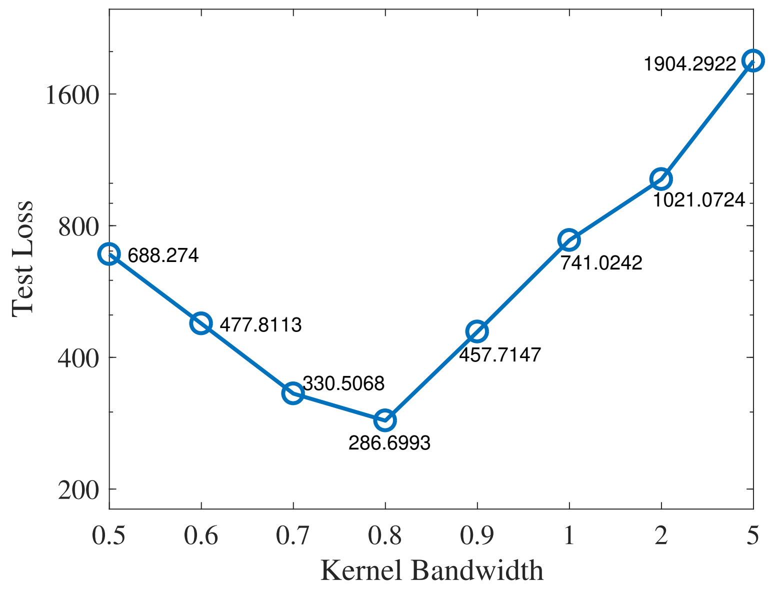


Figure 2.3: Test loss versus smoothing kernel bandwidth on IPTV dataset.

Table 2.3: Average performance with different numbers of anchor points by the global algorithms on IPTV, Yelp, and Reddit datasets.

Datasets	Metrics	Number of Anchor Points				
		1	2	5	10	15
IPTV	MAR	5.175	2.934	1.705	1.667	**1.666**
	MAE	822.1	716.3	620.1	449.0	**383.0**
	Err %	12.67	10.82	9.15	6.44	**5.46**
Yelp	MAR	116.0	106.6	**94.63**	95.74	95.09
	MAE	845.7	805.9	**520.7**	581.7	591.0
	Err %	23.71	22.74	**14.98**	17.34	17.62
Reddit	MAR	49.14	11.50	6.177	6.129	**6.062**
	MAE	8476	8117	6909	6138	**5508**
	Err %	21.50	20.60	17.64	15.88	**14.41**

when the number of anchor points increases. The results also indicate the bottleneck of the performance given enough anchor points. As the (u, i) pairs are sparse in the space, i.e., we only have 4726 (u, i) pairs on a 7100×436 matrix for IPTV data, it is not appropriate to set too many anchor points. Therefore, the number of anchor points should depend on the sparsity of the pairs on user-item dimension. It is also worth mentioning that just a few anchor points, e.g., 5, can render pretty good results.

For Yelp data, as the dataset only has 100 users, it reaches the best performance when the number of anchor points is in the range of $(5 \sim 10)$. However, when the number of anchor points further increases, bias may be introduced as some anchor points are similar, so it actually calculates one type of local neighbors repeatedly in eq. (2.7), which finally lowers the prediction performance. Therefore, selecting anchor points should also depend on the user dimension and the item dimension rather than the pairs' sparsity only.

Effect of Anchor Point Selection

In this section, we investigate some clustering strategies to select a set of anchor points and compare the model performance with the one adopting random anchor point selection. We compare K-Means Clustering and Hierarchical Clustering. K-Means is one of the most popular partitional clustering methods and is computational efficient. Hierarchical Clustering groups the data simultaneously over different scales of distance by creating a multi-level cluster tree, where clusters at a lower level are joined as clusters at the next higher level. Unlike K-Means that produces a single partitioning, Hierarchical Clustering can give different partitions depending on the level-of-resolution.

For the category of Hierarchical Clustering methods, we adopt classic metrics such as "Seuclidean" as shown in Table 2.4 to measure the distance between a pair of data points. The metric definitions are described in $MATLAB$[3]. Specifically, "Cosine" is defined as "one minus the cosine of the included angle between points (treated as vectors)", "Correlation" is defined as "one minus the sample correlation between points", and "Spearman" is defined as "one minus

[3]https://www.mathworks.com/help/stats/pdist.html

Table 2.4: Copenetic correlations of different distance metrics with different methods for computing distance between clusters in Hierarchical Clustering on IPTV dataset.

Metric	Methods for Computing Distance Between Clusters						
	Average	Centroid	Complete	Median	Single	Ward	Weighted
Seuclidean	0.7990	0.7945	0.5810	0.7382	0.7875	0.6527	0.7318
Squaredeuclidean	0.8932	0.8918	0.7220	0.8758	0.8939	0.8687	0.8841
Mahalanobis	0.7605	0.7628	0.4384	0.6913	0.7105	0.4703	0.7086
Cityblock	0.9317	0.9336	0.8150	0.9027	0.9238	0.8639	0.8831
Minkowski	0.9339	**0.9379**	0.7998	0.9187	0.9327	0.8925	0.8946
Chebychev	0.8753	0.8792	0.8274	0.8386	0.8780	0.8113	0.8584
Cosine	0.6431	0.5943	0.5766	0.5566	0.4537	0.5256	0.5952
Correlation	0.6348	0.5997	0.5648	0.5343	0.4449	0.4838	0.5366
Spearman	0.6235	0.5986	0.5677	0.5048	0.3670	0.4950	0.5468

the sample Spearman's rank correlation between observations". In addition, we adopt a set of metrics such as "Average", "Centroid", and "Complete" for computing the distance between clusters. The definitions are described in $MATLAB^4$. Finally, we follow the cluster analysis in $MATLAB^5$ and use the Cophenetic correlation coefficient[6] to verify whether the cluster tree generated from a particular metric is consistent with the pair-wise distances between original data points. Large values (close to 1) indicate high-quality clustering results that capture the pair-wise distances well. We show the Cophenetic correlations of clustering results using the combinations of different metrics for computing the pairwise distance between pairs of points and different metrics for computing the distance between clusters. As shown in Table 2.4, we find that using *Minkowski* distance metric and adopting *centroid* algorithm for computing distance between clusters achieve the best cophenetic correlations for Hierarchical Clustering on IPTV dataset. It is obvious that both *Cityblock* and *Minkowski* with *average*, *centroid*, or *single* algorithm can achieve quite competitive clustering performance.

Finally, we compare the prediction performance of the models with different numbers of anchor points selected by random selection, K-Means Clustering, and Hierarchical Clustering strategies. For K-Means, we adopt the squared Euclidean distance metric for computing pair-

[4]https://www.mathworks.com/help/stats/linkage.html
[5]https://www.mathworks.com/help/stats/examples/cluster-analysis.html
[6]https://www.mathworks.com/help/stats/cophenet.html

Table 2.5: Average performance with different strategies of anchor point selection on IPTV dataset.

Metrics	10 Anchor Points			15 Anchor Points		
	Random	K-Means	Hierarchical	Random	K-Means	Hierarchical
MAR	1.667	2.379	**1.662**	1.666	1.968	1.666
MAE	449.0	502.9	**363.2**	383.0	409.1	386.0
Err %	6.44	7.31	**5.16**	5.46	5.86	5.51

wise distances. For Hierarchical Clustering, we adopt the best strategy based on the cophenetic correlation. As shown in Table 2.5, the prediction performance of the model strongly depends on the anchor selection strategies. Specifically, we find that sometimes several clusters only contain a few (less than 10) members when applying K-Means Clustering on IPTV dataset. In such cases, the imbalance of cluster size may influence the representativeness of selected anchor points, which affects the prediction performance. When using Hierarchical Clustering, the cluster performance is good since we adopt the best strategy relying on the cophenetic correlation, and the model achieves pretty good prediction accuracy with only 10 anchor points, even better than randomly selecting 15 anchor points.

In summary, there are several factors to consider when choosing a proper strategy to select anchor points. A Hierarchical Clustering strategy can improve the model prediction accuracy and decrease the computational cost with relatively fewer anchor points. However, it introduces additional computational cost (e.g.,$O(P^2)$ complexity for K-Means Clustering and $O(P^3)$ for Hierarchical Clustering) in the pre-processing. Selecting sufficiently large number of anchor points randomly may achieve comparable performance but increase model complexity as described in section 2.3.4. In cases where the clustering complexity is negligible comparing to model complexity, a well chosen Hierarchical Clustering strategy is preferred.

Comparison of Global and Parallel Algorithms

We also compare the results of the global and parallel algorithms and present the results in Table 2.6. It is obvious that the parallel algorithm performs better than the global algorithm and achieves similar results with a smaller number of anchor points. The reason is that the parallel algorithm is more flexible in controlling the nuclear norm for parameter matrices in comparison

Table 2.6: Average performance with different numbers of anchor points by global and parallel algorithms on IPTV dataset.

	Metrics	Number of Anchor Points				
		1	2	5	10	15
Global	MAR	5.175	2.934	1.705	1.667	**1.666**
	MAE	822.1	716.3	620.1	449.0	**383.0**
	Err %	12.67	10.82	9.15	6.44	**5.46**
Parallel	MAR	5.136	2.865	1.684	1.678	**1.676**
	MAE	822.2	713.7	486.3	379.0	**362.7**
	Err %	12.33	10.62	7.06	5.40	**5.16**

with the global algorithm, which only assumes the combination of a number of block matrices low-rank. Specifically, for the global algorithm, only three parameters (λ,β,ρ) are used to control the nuclear norm of model parameter X. For the parallel algorithm, there are up to $3 \cdot q$ parameters in total and each tuple (λ_τ,β_τ,ρ_τ) can be used to control the rank for each local model. Therefore, the parallel algorithm is more compatible with the local low-rank assumption when dealing with the nuclear norm. In the experiments, however, we find that it only slightly improves the results, so we choose the same parameters for all local models with the nuclear norm. Meanwhile, we can see that the prediction accuracy will converge as the number of anchor points grows.

CHAPTER 3
GEOMETRIC HAWKES PROCESSES

3.1 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 [89], inferring Granger causality [87], and predicting recurrent user behaviors [97, 19, 74]. 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 [20] 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 [56] 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 [24, 59, 91], which captures the correlations of both endogenous and exogenous event intensities. Extensive studies [54, 23, 22, 87] 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 [22, 32].

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 [8, 18, 44, 65] 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 chapter, 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.

3.2 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 3.1.

3.2.1 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.

Table 3.1: 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

The conditional intensity function that characterizes a Hawkes process is defined as:

$$\lambda(t) = \eta + \alpha \sum_{t_i \in \mathcal{T}_t} \kappa(t - t_i), \tag{3.1}$$

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())*:

$$\kappa(t) = Zero() = 0, \tag{3.2}$$

Linear (Linear(a,b)):

$$\kappa(t) = Linear(a, b) = a(1 - \frac{b}{a}t) \tag{3.3}$$

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

$$\kappa(t) = EXP(a, b) = ae^{-bt}. \tag{3.4}$$

Power-Law (PWL(a, c, p): The power-low kernel is usually used for modeling a slower rate of decay than exponential [2]:

$$\kappa(t) = PWL(a, c, p) = \frac{a}{(t+c)^p}. \tag{3.5}$$

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 [30]:

$$\kappa(t) = Qexp(a, q) = \begin{cases} ae^{-t}, & q = 1 \\ a[1+(q-1)t]^{\frac{1}{1-q}}, & q \neq 0 \text{ and } 1+(1-q)t > 0 \\ 0, & q \neq 0 \text{ and } 1+(1-q)t \leq 0. \end{cases} \tag{3.6}$$

Rayleigh (Ray(a, b)): The Rayleigh kernel has been used for modeling a non-monotonically decaying effect [70]:

$$\kappa(t) = Ray(a, b) = ate^{-bt^2} \tag{3.7}$$

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:

$$\lambda_u(t) = \boldsymbol{h}_u + \boldsymbol{a}_u \sum_{t_j^u \in \mathcal{T}_t^u} \kappa(t - t_j^u), \tag{3.8}$$

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:

$$\lambda_{(u,i)}(t) = \boldsymbol{H}_{u,i} + \boldsymbol{A}_{u,i} \sum_{t_j^{u,i} \in \mathcal{T}_t^{u,i}} \kappa(t - t_j^{u,i}), \tag{3.9}$$

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.

3.2.2 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. (3.10), the random walk normalized Laplacianeq. (3.11), and the symmetric normalized Laplacianeq. (3.12):

$$L^c = D - W, \tag{3.10}$$

$$L^{rw} = D^{-1}L^{c} \tag{3.11}$$

$$L^{sys} = D^{-1/2}L^{c}D^{-1/2} = I_n - D^{-1/2}WD^{-1/2}, \tag{3.12}$$

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 [58]. 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 $\boldsymbol{x} = U\hat{\boldsymbol{x}}$ [76]. Thus, the spectral convolutional of function $\boldsymbol{x}$ and convolutional kernel function $\boldsymbol{y}$ on graph G is given by [8]:

$$(\boldsymbol{x} \star \boldsymbol{y})_G = U \cdot diag([\hat{y}(\lambda_0), ..., \hat{y}(\lambda_{n-1})]) \cdot U^{\top}\boldsymbol{x}, \tag{3.13}$$

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 [58]. 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 [33], a polynomial filter was introduced by [18]. 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 [33], 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 [18]:

$$\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\boldsymbol{x}), \tag{3.14}$$

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 $\mathcal{O}(n^2)$ [18], 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 [44]. For example, assuming $K = 2$ and $\lambda_{max} = 2$, we can get the first-order model as:

$$\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{1} \theta_k T_k(L - I)\boldsymbol{x}) = \sigma((\theta_0 - \theta_1 D^{-1/2} W D^{-1/2})\boldsymbol{x}). \tag{3.15}$$

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:

$$\boldsymbol{x}_{output} = \sigma(\theta(I + D^{-1/2} W D^{-1/2})\boldsymbol{x}), \tag{3.16}$$

A even more simplified approximation model can be obtained through a re-normalization trick [44]:

$$\boldsymbol{x}_{output} = \sigma(\theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} \boldsymbol{x}). \tag{3.17}$$

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 [47, 65]. 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 [65]:

$$\boldsymbol{X}_{output} = \sigma(\sum_{k=0}^{K-1} \sum_{k'=0}^{K-1} \theta_{kk'} T_k(\tilde{L}_r) \boldsymbol{X} T_{k'}(\tilde{L}_c)), \tag{3.18}$$

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 [13] and convolution implementation in Caffe [40]. 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:

$$\boldsymbol{x}_{c'_{output}} = \sigma(\sum_{c=1}^{C} \sum_{k=0}^{K-1} \theta_{kc,c'} T_k(\tilde{L}) \boldsymbol{x}_c). \tag{3.19}$$

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:

$$\boldsymbol{X}_{c'_{output}} = \sigma(\sum_{c=1}^{C}\sum_{k=0}^{K-1}\sum_{k'=0}^{K-1} \theta_{kk'c,c'} T_k(\tilde{L}_r)\boldsymbol{X}_c T_{k'}(\tilde{L}_c)). \tag{3.20}$$

It is straightforward to expand the eqs. (3.15) to (3.17) to the generalized multi-GCN layer version. For example, for re-normalization trick model eq. (3.17):

$$\boldsymbol{X}_{c'_{output}} = \sigma(\sum_{c=1}^{C} \theta_{c,c'} \tilde{D}_r^{-1/2}\tilde{W}_r\tilde{D}_r^{-1/2}\boldsymbol{X}_c\tilde{D}_c^{-1/2}\tilde{W}_c\tilde{D}_c^{-1/2}), \tag{3.21}$$

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 [65]. 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 [38]. 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 $\boldsymbol{X}^{(T)} = \boldsymbol{X}^{(T-1)} + \boldsymbol{dX}^{(T-1)}$.

3.2.3 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. (3.8) 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. (3.9), 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. (3.14) and eq. (3.18) 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 [1], 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. (3.8) and eq. (3.14), 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:

$$\mathcal{L}(\mathcal{T}^u \mid \boldsymbol{x_{\theta,\zeta}}^{(T)}) = \sum_{t_j^u \in \mathcal{T}^u} \log(\boldsymbol{x}_u^{(T)} \Phi_j^u) - \boldsymbol{x}_u^{(T)} \Psi^u, \tag{3.22}$$

where:

$$\begin{aligned} \boldsymbol{x}_u^{(T)} =& (\boldsymbol{h}(u)^{(T)}, \boldsymbol{a}(u)^{(T)}), \\ \Phi_j^u =& (1, \sum_{t_k^u < t_j^u} \kappa(t_j^u - t_k^u))^\top, \\ \Psi^u =& (\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \int_{t_j^u}^{\Gamma} \kappa(t - t_j^u)dt)^\top. \end{aligned} \tag{3.23}$$

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. (3.1), (3.8) and (3.9), we set $a = 1$ in eqs. (3.3) to (3.7). When adopting zero kernels, the second term of the feature vector Φ_j^u and the integral Ψ^u becomes zero, by integrating eq. (3.2) into eq. (3.23):

$$\begin{aligned}\Phi_j^u =&(1, 0)^\top,\\ \Psi^u =&(\Gamma, 0)^\top,\end{aligned} \tag{3.24}$$

When adopting linear kernels, the vectors can be computed by integrating eq. (3.3) into eq. (3.23):

$$\begin{aligned}\Phi_j^u =&(1, \sum_{t_k^u < t_j^u} [1 - b(t_j^u - t_k^u)])^\top,\\ \Psi^u =&(\Gamma, \sum_{t_j^u \in \mathcal{T}^u} [\frac{b}{2}(\Gamma - t_j^u)^2 - (\Gamma - t_j^u)])^\top,\end{aligned} \tag{3.25}$$

For exponential kernels, the vectors can be computed by integrating eq. (3.4) into eq. (3.23):

$$\begin{aligned}\Phi_j^u =&(1, \sum_{t_k^u < t_j^u} e^{-b(t_j^u - t_k^u)})^\top,\\ \Psi^u =&(\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{b}(1 - e^{-b(\Gamma - t_j^u)}))^\top,\end{aligned} \tag{3.26}$$

For power-law kernels, the vectors can be computed by integrating eq. (3.5) into eq. (3.23):

$$\begin{aligned}\Phi_j^u =&(1, \sum_{t_k^u < t_j^u} \frac{1}{(t_j^u - t_k^u + c)^p})^\top,\\ \Psi^u =&(\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{p-1}[c^{1-p} - (\Gamma - t_j^u + c)^{1-p}])^\top,\end{aligned} \tag{3.27}$$

For Tsallis Q-exponential kernels with $1 < q < 2$, the vectors can be computed by integrating integrating eq. (3.6) into eq. (3.23):

$$\begin{aligned}\Phi_j^u =&(1, \sum_{t_k^u < t_j^u} [1 + (q-1)(t_j^u - t_k^u)]^{\frac{1}{1-q}})^\top,\\ \Psi^u =&(\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{2-q}\{1 - [1 + (q-1)(\Gamma - t_j^u)]^{\frac{2-q}{1-q}}\})^\top,\end{aligned} \tag{3.28}$$

For Rayleigh kernels, the vectors can be computed by integrating integrating eq. (3.7) into eq. (3.23)

$$\begin{aligned}\Phi_j^u =&(1, \sum_{t_k^u < t_j^u} (t_j^u - t_k^u) e^{-b(t_j^u - t_k^u)^2})^\top,\\ \Psi^u =&(\Gamma, \sum_{t_j^u \in \mathcal{T}^u} \frac{1}{2b}(1 - e^{-b(\Gamma - t_j^u)^2}))^\top,\end{aligned} \tag{3.29}$$

Algorithm 3: Algorithm for Learning single-graph GHP

```
Input: All the training events $\mathcal{O} = \{\mathcal{T}^u\}_u$; parameters $\rho$, $\gamma$, $\beta$; $\{\boldsymbol{x}_c = [\boldsymbol{h}_c; \boldsymbol{a}_c]\}_{c=1}^C$
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. (3.19) 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. (3.30) and
           update weights $\boldsymbol{\theta}$, $\boldsymbol{\zeta}$
    end
    Output $\{\boldsymbol{x}_c^{(T)}\}_{c=1}^C$ to calculating Hawkes intensity by eq. (3.8).
end
```

It is worth mentioning that the notation ${\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)}$ 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 $h(\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}) = \rho\{tr(\boldsymbol{h}^\top L\boldsymbol{h}) + tr(\boldsymbol{a}^\top L\boldsymbol{a})\}$ and the squared Frobenius norm $g(\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}) = \gamma\|\boldsymbol{h}\|_F^2 + \beta\|\boldsymbol{a}\|_F^2$ as [69]. Finally, we can obtain $\boldsymbol{h}$ and $\boldsymbol{a}$ by minimizing the following objective function:

$$OPT = \min_{\boldsymbol{\theta},\boldsymbol{\zeta}} - \frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^u \in \mathcal{O}} \mathcal{L}(\mathcal{T}^u \mid {\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)}) + h({\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)}) + g({\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)}), \quad s.t.\ {\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}}}^{(T)} \geq \mathbf{0}, \tag{3.30}$$

where $\boldsymbol{x}_{\boldsymbol{\theta},\boldsymbol{\zeta}} = [\boldsymbol{h}; \boldsymbol{a}]$, and $\rho\, \gamma$, β control the trade-off between the constrains. After the parameters converging to optimal, we can directly use $\boldsymbol{x}$ and eq. (3.8) to compute the intensity and make predictions.

Multi-graph GHP

Similarly, we can give the objective function of multi-graph GHP. According to eq. (3.9) and eq. (3.18), 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:

$$\mathcal{L}(\mathcal{T}^{u,i} \mid {\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}}^{(T)}) = \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}_{u,i}^{(T)} \Phi_j^{u,i}) - \boldsymbol{X}_{u,i}^{(T)} \Psi^{u,i}, \tag{3.31}$$

where:

$$\begin{aligned}
\boldsymbol{X}_{u,i}^{(T)} &= (\boldsymbol{H}(u,i)^{(T)}, \boldsymbol{A}(u,i)^{(T)}), \\
\Phi_j^{u,i} &= (1, \sum_{t_k^{u,i} < t_j^{u,i}} \kappa(t_j^{u,i} - t_k^{u,i}))^\top, \\
\Psi^{u,i} &= (\Gamma, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{\Gamma} \kappa(t - t_j^{u,i}) dt)^\top.
\end{aligned} \tag{3.32}$$

Note that the feature vector $\Phi_j^{u,i}$ and the integral $\Psi^{u,i}$ can be calculated in a way similar to eq. (3.23). Thus, we omitted the closed forms of different kernels.

```
Algorithm 4: Algorithm for Learning multi-graph GHP
  Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; parameters $\rho$, $\gamma$, $\beta$; $\{\boldsymbol{X}_c = [\boldsymbol{H}_c; \boldsymbol{A}_c]\}_{c=1}^C$
  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. (3.20) 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. (3.33) and
           update weights $\boldsymbol{\Theta}$, $\boldsymbol{\zeta}$
      end
      Output $\{\boldsymbol{X}_c^{(T)}\}_{c=1}^C$ to calculating Hawkes intensity by eq. (3.9).
  end
```

In multi-graph case, the notation $\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}{}^{(T)}$ 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 $\tilde{h}(\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}) = \rho\{tr(\boldsymbol{H}^\top L_r \boldsymbol{H}) + tr(\boldsymbol{H} L_c \boldsymbol{H}^\top) + tr(\boldsymbol{A}^\top L_r \boldsymbol{A}) + tr(\boldsymbol{A} L_c \boldsymbol{A}^\top)\}$ [41]. 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 $\tilde{g}(\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}) = \gamma\|\boldsymbol{H}\|_* + \beta\|\boldsymbol{A}\|_*$ as [20], 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:

$$OPT = \min_{\boldsymbol{\Theta},\boldsymbol{\zeta}} -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}}\mathcal{L}(\mathcal{T}^{u,i}|\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}{}^{(T)}) + \tilde{h}(\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}{}^{(T)}) + \tilde{g}(\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}{}^{(T)}), \quad s.t.\ \boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}}{}^{(T)} \geq \boldsymbol{0}, \quad (3.33)$$

where $\boldsymbol{X}_{\boldsymbol{\Theta},\boldsymbol{\zeta}} = [\boldsymbol{H};\boldsymbol{A}]$, 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. (3.9) to compute the intensity and make predictions.

Learning with Clipping

We can use several stochastic optimization algorithms such as SGD and Adam [43] 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. (3.30) and (3.33). Since it is the inequality constraints, directly solving it by adding Lagrange multiplier or Kuhn-Tucker method [81] 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 Algorithm 3 and Algorithm 4 for single-graph and multi-graph GHP, respectively.

Computational Complexity

By applying polynomial localization, the single-GCN eq. (3.14) reaches $\mathcal{O}(n)$ [18] rather than using eq. (3.18) with complexity $\mathcal{O}(n^2)$, where n is the number of vertices of the graph. Thus, the multi-GCN has the complexity of $\mathcal{O}(mn)$ [65] 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 $\boldsymbol{W} = 4n_c^2 + 4n_cn_i + n_cn_o + 3n_c$ [72], 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.

3.3 Experiment and Results

In this section, we introduce the experiments.

3.3.1 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 2.2. Specifically, the **IPTV** dataset [87] 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 **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 [65], 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

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

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. (3.30) and (3.33) 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 [84], we report *mean average rank* (MAR) of all testing cases. A smaller value of MAR indicates better predictive performance.

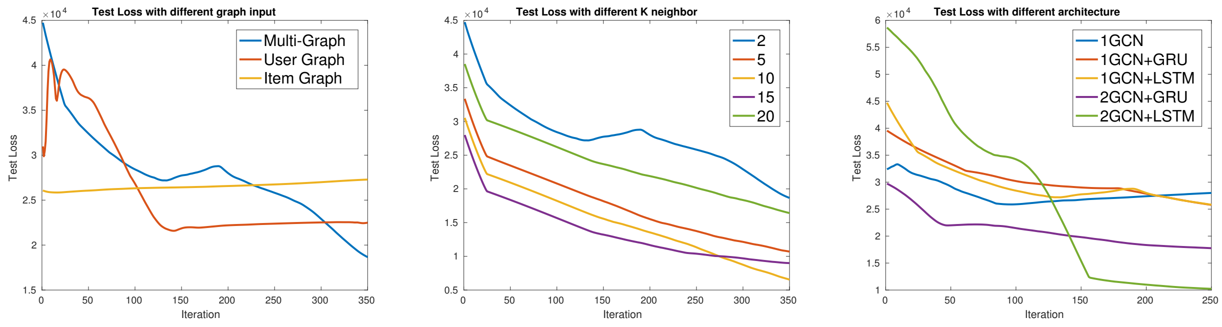
Time Prediction Accuracy: Given a specific pair of user u and the item i, we record the *mean absolute error* (MAE) of the next predicted time and the ground truth of testing time t. The predicted time is calculated by the density of next event time as $f(t) = \lambda_{(u,i)}(t) S_{(u,i)}(t)$, and then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

3.3.2 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 [14]. 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 [20] assuming that all processes are independent and the parameters are low rank matrices. However, there are no interactions



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

Figure 3.1: Testing loss with respect to different graph inputs, different number of neighbors, and architectures on IPTV data.

between different processes.

Coevol: Coevolve is a co-evolutionary latent feature process [84] which constructs interdependent Hawkes processes by embedding user and item features globally into each process. This method actually combines all events happening 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.

3.3.3 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. 3.1(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. 3.1(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. 3.1(a) and fig. 3.1(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 [20].

Variations of Architecture Setting

We compare different architecture settings of our model and the results are presented in fig. 3.1(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 [15] 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 [44, 65]. 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

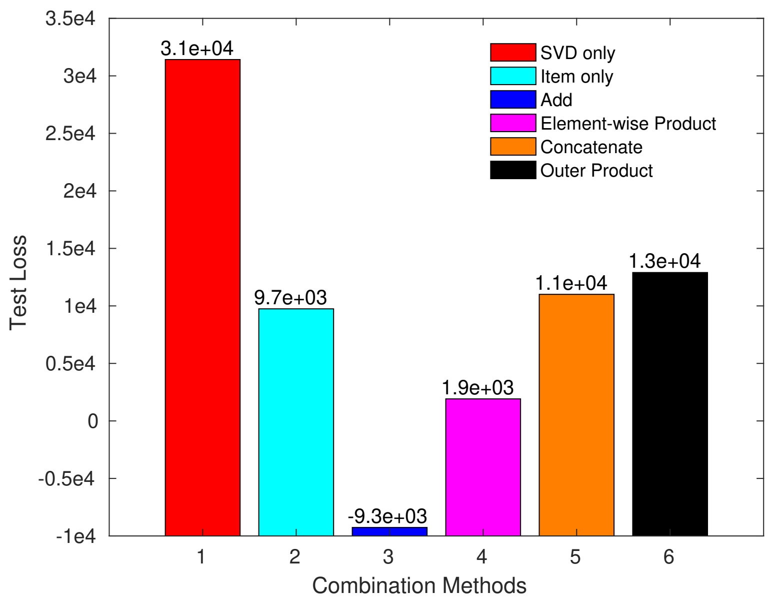


Figure 3.2: Test loss with respect to different combination methods of item features on IPTV dataset.

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 in fig. 3.2. 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

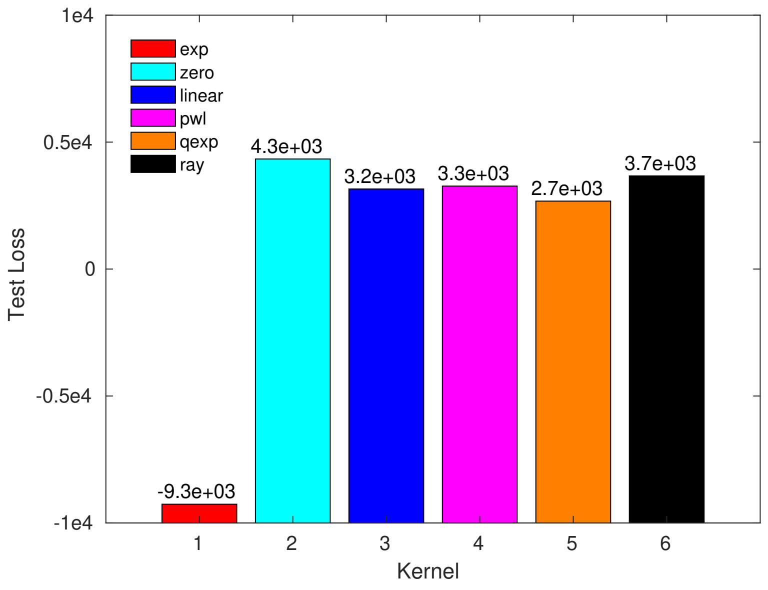


Figure 3.3: Test loss with respect to different kernels on IPTV dataset.

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. (3.4) to (3.7). As shown in fig. 3.3, 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 [70, 19].

Different Graph Propagation Models

We compare the test loss with different graph propagation models [44] on the IPTV dataset. The results shown in table 3.2 indicated that the first order term only model and the Chebyshev filter eq. (3.14) with $K = 2$ both achieve comparable performance than other graph propagation models eqs. (3.15) to (3.17). 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 exper-

Table 3.2: Test loss with respect to different graph propagation models on IPTV dataset.

Description	Propagation Model	Test Loss
$K = 2$ Chebyshev filter eq. (3.14)	$\boldsymbol{x}_{output} = \sigma(\sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\boldsymbol{x})$	-9.26e+03
$K = 3$ Chebyshev filter eq. (3.14)		-7.11e+03
1st-order model eq. (3.15)	$\boldsymbol{x}_{output} = \sigma((\theta_0 - \theta_1 D^{-1/2} W D^{-1/2})\boldsymbol{x})$	-5.04e+03
Single parameter 1st-order model eq. (3.16)	$\boldsymbol{x}_{output} = \sigma(\theta(I + D^{-1/2} W D^{-1/2})\boldsymbol{x})$	-3.14e+03
Re-normalization trick eq. (3.17)	$\boldsymbol{x}_{output} = \sigma(\theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} \boldsymbol{x})$	-2.37e+03
1st-order term only	$\boldsymbol{x}_{output} = \sigma(\theta D^{-1/2} W D^{-1/2} \boldsymbol{x})$	-9.69e+03

Table 3.3: 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

iments, we conclude that the graph Lapacian which depends on the graph construction and the graph convolution network structures are crucial to the performance of our GHP model.

3.3.4 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 3.3. 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 [84] 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.

CHAPTER 4
FAIRNESS-AWARE GRAPH CONVOLUTIONAL POINT PROCESSES

4.1 Introduction

Human events are often recurrent and exhibit self-exciting properties. For example, in patient diagnosis events, elevated risk exists for a patient that has been recently at risk. In TV watching events, after watching an episode drama at a certain time, users are likely to watch the remaining episodes daily or weekly in the following days. Time intervals between consecutive events carry rich information about specific types of user-item interaction events. Point processes, which assume certain forms of inter-event correlations, have been widely used in predicting temporal events such as online check-ins, job-seeking events, and electronic records of hospital admissions [19, 83, 89].

A joint modeling of point processes with coevolving user and item interests has been shown more effective for capturing the temporal dynamics [25, 84, 79]. On one hand, the instantaneous rate of interaction event occurrence depends on user interests and item profiles, which may evolve over time. On the other hand, the interactions between users and items may also drive the evolution of user interests and item features as reflected by intuitive examples such as "you are what you read and what you tweet". Specifically, each user/item is associated with a dynamic changing feature in a latent space learned from basic features such as demographics, the features of items he or she interacted with, and user-item interaction features such as reviews. The temporal correlations between interaction events with coevolutionary dynamics are captured through high dimensional intertwined stochastic processes.

Despite promising examples of point processes with coevolutionary dynamics, there are still critical research challenges due to the data sparsity and imbalances with respect to certain user/item groups. Firstly, the coevolving dynamics driven by the occurrence of user-item interaction would fail to achieve good performance when there are insufficient interaction events for each process. In fact, a large portion of users and items have very few interaction events in many applications. For example, Netflix movie rating dataset contains 480 k users and 17 k items and most users only rate a few movies. In the LinkedIn dataset of 2,439 users and 82 items, only 12.8% have more than 4 reported job histories. Secondly, unfair predictions may be generated due to data bias and can be amplified through self-excitation. For example, job-seeking event prediction based on interaction data may exhibit socioeconomic bias since users with frequent interaction events on LinkedIn tend to be the communities of higher socioeconomic status.

To tackle these challenges, we present a novel fairness-aware graph convolutional point processes with coevolutionary feature embedding to strike a balance between the accuracy and fairness in this chapter. Our major contributions are: (1) We learn the joint modeling of point processes under geometric structure with coevolutionary feature embedding to achieve higher prediction accuracy especially for processes with insufficient observations. (2) We present some novel fairness metrics for introducing user and item parity into temporal event prediction. By adding penalty terms of fairness costs to the likelihood function, our model enforces that the expected intensity over a time period for each of the user/item groups to be equal. To our best knowledge, our model is the first one that introduces fairness in point processes to model user-item interactions. (3) We propose an efficient convex optimization algorithm to estimate model parameters and discuss both scalability and efficiency of the algorithm. Extensive experiments on real world datasets demonstrate that our method improves event prediction over baselines and controls the balance between accuracy and fairness effectively.

4.2 Background

We first introduce background knowledge and list key notations in Table 4.1.

4.2.1 Point Process

Point processes are applied to modeling the temporal dynamics observed in event sequences (e.g., TV browsing events) between of a group of m users and n items (e.g., TV programs). Instead of discretizing the time into bins as traditional methods [31, 37, 83], the time of each interaction event is modeled as a random variable. Formally, the ordered list of all observed q events are denoted as $\mathcal{O} = \{e_k = (u_k, i_k, t_k)\}_{k=1}^{q}$ in a window $[0, T]$, where $u_k \in \{1, ..., m\}$, $i_k \in \{1, ..., n\}$, $t_k \in \mathbb{R}^+$, $0 \leq t_1 \leq t_2 ... \leq t_q \leq T$. The event sequence (e.g., browsing

Table 4.1: Key notations.

Variable	Description
L	Laplacian matrix of graph
D	degree matrix of graph
W	adjacency matrix of graph
Λ	diagonal eigenvalue matrix
U	orthonormal eigenvectors matrix
$\tilde{\Lambda}$	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
$\mathcal{G}_u, \mathcal{G}_i$	user graph and item graph
m, n	number of nodes in $\mathcal{G}_u, \mathcal{G}_i$
u, i	the u-th and i-th entry of user and item
$\tilde{D}_{\mathcal{G}_u}, \tilde{D}_{\mathcal{G}_i}$	renormalized degree matrix of user and item graph
$\tilde{W}_{\mathcal{G}_u}, \tilde{W}_{\mathcal{G}_i}$	renormalized adjacency matrix of user and item graph
$\mathbf{\Theta}$	graph convolutional filter parameter matrix
$\mathbf{\Theta}_k$	k-th polynomial filter parameter matrix in $\mathbf{\Theta}$
$\mathbf{\Theta}_{\mathcal{G}_u}, \mathbf{\Theta}_{\mathcal{G}_i}$	coefficient matrices of user and item process
$\mathbf{\Omega}$	product of the graph covolutional filter matrices
$\mathcal{T}$	a list of discrete temporal events
$\mathcal{O}$	observed events of all user-item pairs
T	total time in all the sequences
e_k	k-th event in $\mathcal{O}$
u_k, i_k	the specific u and i for e_k
t_k	the time when e_k happened for specific (u_k, i_k) pair
$\lambda^{u,i}(t)$	point process intensity function for (u, i) pair
$\mathbf{\Lambda}(t)$	point process intensity function matrix
$\kappa_\omega(t)$	kernel function in point process with bandwidth ω
$\eta^{u,i}$	base intensity in point process for (u, i) pair
$\alpha^{u,i}$	self-exciting coefficient in Hawkes process for (u, i) pair
$\boldsymbol{H}$	point process base intensity matrix
$\hat{\boldsymbol{H}}$	auxiliary variable matrix for alternating updates
c, d	input and output channel dimensions
α, ρ	trade-off parameters for constraints
γ, β	trade-off parameters for fairness
$\boldsymbol{X}_u(t), \boldsymbol{X}_i(t)$	base drift of the user and item feature
$\boldsymbol{u}_u(t), \boldsymbol{i}_i(t)$	dynamic coevolved feature for user and item
$\boldsymbol{y}_u(t), \boldsymbol{z}_i(t)$	d-dimensional point process for user u and item i
$\boldsymbol{U}(t), \boldsymbol{I}(t)$	dynamic convolved user and item feature matrix
$\boldsymbol{Y}_{\mathcal{G}_u}(t), \boldsymbol{Z}_{\mathcal{G}_i}(t)$	collections of user and item point processes
$\tilde{\lambda}$	intensity integral over time after normalization

history) of each user-item pair (u, i) can be modeled as a univariate process and the process can be characterized via its conditional intensity $\lambda^{u,i}(t)$, which is the expected rate of the event occurrences for the user-item pair given the history of all the previous events up to time t.

Different types of point processes assume certain forms of dependency on the history in various ways. For example, the Poisson process makes the assumption that the duration is stationary and the intensity function $\lambda^{u,i}(t) = \eta^{u,i}$, where $\eta^{u,i}$ is constant for a specific (u, i) pair over time. The Hawkes process [35] is an extension of Poisson process which add a exponential decay of history events. It can be presented as:

$$\lambda^{u,i}(t) = \eta^{u,i} + \alpha^{u,i} \sum_{t_k^{u,i} < t} \kappa_\omega(t - t_k^{u,i}), \tag{4.1}$$

where $\eta^{u,i} \geq 0$ is the baseline intensity that captures the long-term tendency of event occurrence, $\alpha^{u,i} \geq 0$ is the self-exciting coefficient of the history decay and $\kappa_\omega = \exp(-t/\omega)$ is the kernel function capturing temporal dependencies, which is usually an exponential kernel with bandwidth ω.

4.2.2 Geometric Deep Learning

Given an undirected weighted graph denoted as $G = (V, \mathcal{E}, W)$, where V is a finite set of $|V| = n$ vertices, $\mathcal{E}$ is the set of edges and $W \in \mathbb{R}^{n\times n}$ is the adjacency matrix with entries $W_{ij} > 0$ if $(i, j) \in \mathcal{E}$, an $n \times n$ symmetric positive-semidefinite matrix which is called the graph Laplacian can be constructed to reflect useful properties of the graph. Usually, the normalized graph Laplacian can be denoted as:

$$L = I_n - D^{-1/2} W D^{-1/2}, \tag{4.2}$$

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.

In order to extract meaningful patterns from graph, the graph convolution is proposed in the spectral domain [8, 36]. Specifically, suppose $\boldsymbol{x} \in \mathbb{R}^n$ is a signal of graph G, where n is the number of nodes in the graph, then the spectral graph convolution operator $\star$ is defined as follows:

$$\boldsymbol{x} \star \boldsymbol{y} = U((U^\top \boldsymbol{x}) \odot (U^\top \boldsymbol{y})) = U diag(U^\top \boldsymbol{y}) U^\top \boldsymbol{x} = U g_{\boldsymbol{\theta}}(\Lambda) U^\top \boldsymbol{x}, \tag{4.3}$$

where $\odot$ is the element-wise Hadamard product, $L = U \Lambda U^\top$ is the normalized graph Laplacian of G, U is the matrix of orthonormal eigenvectors of L, and Λ is the diagonal matrix with the non-negative eigenvalues of L, $g_{\boldsymbol{\theta}}(\Lambda) = diag([\hat{y}(\lambda_0), ..., \hat{y}(\lambda_{n-1})])$, and $\boldsymbol{\theta} \in \mathbb{R}^n$ is the learnable convolutional filters. For simplicity, we denote $\boldsymbol{x} \star \boldsymbol{y}$ as $g_{\boldsymbol{\theta}} \star \boldsymbol{x}$.

For the purpose of reducing the computational complexity and the parameters in the model, as well as incorporating localization following the lead of graph signal processing [33], a polynomial filter was introduced by [18]:

$$g_{\boldsymbol{\theta}}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), \tag{4.4}$$

where $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$ is the rescaled eigenvalues in the interval $[-1, 1]$, $T_k(\tilde{\Lambda}) = 2\tilde{\Lambda} \odot T_{k-1}(\tilde{\Lambda}) - T_{k-2}(\tilde{\Lambda})$ is defined in a recursive way with $T_0 = I$ and $T_1 = \tilde{\Lambda}$. As $\boldsymbol{\theta} = \{\theta_k\}_{k=0}^{K}$ is a vector of polynomial coefficients for such a filter, the computational complexity of the entire filtering operation becomes $\mathcal{O}(K|\mathcal{E}|)$ rather than $\mathcal{O}(n^2)$ [18] by applying K-localized polynomial kernel, where $|\mathcal{E}|$ is the number of non-zero values in L of the graph G. Then we have:

$$g_{\boldsymbol{\theta}} \star \boldsymbol{x} = \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) \boldsymbol{x}, \tag{4.5}$$

where $\tilde{L} = 2L/\lambda_{max} - I_n$ is the rescaled Laplacian. Note that the formula involves only the computation of the Laplacian L without the computation of its eigendecomposition, and the convolutional kernel with spatial localization will benefit local feature extraction.

A simplified variant of this filter is proposed in [44], which assumes $K = 1$ and $\lambda_{max} = 2$:

$$g_{\boldsymbol{\theta}} \star \boldsymbol{x} = \theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} \boldsymbol{x}, \tag{4.6}$$

with a single parameter constraining $\theta = \theta_0 = -\theta_1$, where $\tilde{W} = W + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{W}_{ij}$. Then the generalized version with c input channels in n-node signal $\boldsymbol{X} \in \mathbb{R}^{n \times c}$ and d filters or feature maps is defined as follows:

$$\boldsymbol{Z} = \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} \boldsymbol{X} \boldsymbol{\Theta}, \tag{4.7}$$

with total complexity $\mathcal{O}(|\mathcal{E}|dc)$, where $\boldsymbol{\Theta} \in \mathbb{R}^{c \times d}$ is the matrix of filter parameters and $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$ is the convolved signal matrix.

4.3 Model

In this section, we introduce our Fairness-aware Graph Convolutional Point Processes. Our model is inspired by techniques like matrix factorization and factorization machine, which shows superior performance over sparsity data. We assume the self exiting component of the Hawkes intensity function eq. (4.1) depends on the interaction of two multivariate point processes. Specifically, the intensity function for a (u, i) pair is characterized as:

$$\lambda^{u,i}(t) = \eta^{u,i} + \boldsymbol{y}_u(t) \boldsymbol{z}_i(t)^{\top}. \tag{4.8}$$

where $\eta^{u,i} \geq 0$ is the baseline intensity, $\boldsymbol{y}_u(t) \in \mathbb{R}^d$ and $\boldsymbol{z}_i(t) \in \mathbb{R}^d$ are the intensity functions for the user process and the item process both with latent d dimensions. The intensity functions are graph convolutional functions defined on user and item features, which may change over time. The convolutional filters can be viewed as the coefficients of the our Graph Convolutional Point Process. The inner product of the user process and the item process captures the compatibility between a user u and an item i.

In the following sections, we first introduce the construction of user and item features in section 4.3.1, and then introduce the details of our Graph Convolutional Point Process in section 4.3.2. In addition, we introduce a set of novel fairness metrics in section 4.3.3, and the final objective function in section 4.3.4.

4.3.1 Dynamic Feature Coevolution

We assume each user or item is associated with a c-dimensional feature vector that changes over time. The dynamic feature of a user can be characterized by both user's basic features and the features of items he or she recently interacted with, which is called "coevolution". For each user u, the corresponding embedding after user u's k-th event $e_k^u = (i_k^u, t_k^u)$ can be formulated as:

$$\boldsymbol{u}_u(t) = \boldsymbol{X}_u(t) + \sum_{e_k^u \in \mathcal{O}^u, t_k^u < t} \kappa_\omega(t - t_k^u)\boldsymbol{i}_{i_k}(t_k^u). \tag{4.9}$$

Similarly, for each item i, we specify $\boldsymbol{i}_i(t)$ as:

$$\boldsymbol{i}_i(t) = \boldsymbol{X}_i(t) + \sum_{e_k^i \in \mathcal{O}^i, t_k^i < t} \kappa_\omega(t - t_k^i)\boldsymbol{u}_{u_k}(t_k^i), \tag{4.10}$$

where $\boldsymbol{X}_u(t) \in \mathbb{R}^c$ and $\boldsymbol{X}_i(t) \in \mathbb{R}^c$ are the *base drift* of the corresponding user and item and the second summation term is the *coevolution* feature averaging. Specifically, the base drifts $\boldsymbol{X}_u(t)$ and $\boldsymbol{X}_i(t)$ are defined based on the features of specific users or items. It can be static or weighted average of these static features observed at different times. The feature averaging processes are modeled by an exponential kernel function $\kappa_\omega(t) = \exp(-\omega t)$ with bandwidth ω, and it reduces the influence of each previous feature.

It can be shown by induction that the feature functions defined in eq. (4.9) and eq. (4.10) can be simplified as the following:

$$\boldsymbol{u}_{u_k}(t_k) = \sum_{j=1}^{k} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{X}_{u_j}(t_j) + \sum_{j=1}^{k-1} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{X}_{i_j}(t_j), \tag{4.11}$$

$$\boldsymbol{i}_{i_k}(t_k) = \sum_{j=1}^{k} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{X}_{i_j}(t_j) + \sum_{j=1}^{k-1} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{X}_{u_j}(t_j), \tag{4.12}$$

where $\mathbb{I}[\cdot]$ is the indicator function. By eq. (4.11) and eq. (4.12), the embedding features can be exactly computed given the base drift features $\boldsymbol{X}_u(t)$ and $\boldsymbol{X}_i(t)$. Once an event of user u_k interacting with item i_k occurs, the corresponding features $\boldsymbol{u}_{u_k}(t_k)$ and $\boldsymbol{i}_{i_k}(t_k)$ will be updated, respectively.

4.3.2 Graph Convolutional Point Processes

As we can see, the feature embedding of users and items are driven by the occurrence of user-item interaction as shown in eq. (4.11) and eq. (4.12). However, the feature embedding may be inaccurate when there are insufficient interaction events for each point process. Thus, we propose to incorporate additional geometric structure in the form of user and item graphs into dynamic feature coevolution for improving temporal prediction accuracy.

We assume that similar users may share similar interaction patterns (e.g., TV browsing behavior). The similarity of users and items can be represented by a user graph and an item graph, respectively. Additional graph information can be used in characterizing point process intensities to enforce intensity smoothness over a graph. Specifically, we apply a graph convolution defined in eq. (4.7) to the user and item features defined in eq. (4.11) and eq. (4.12) to generate intensities for user and item processes:

$$\boldsymbol{Y}_{\mathcal{G}_u}(t) = \tilde{D}_{\mathcal{G}_u}^{-1/2}\tilde{W}_{\mathcal{G}_u}\tilde{D}_{\mathcal{G}_u}^{-1/2}\boldsymbol{U}(t)\boldsymbol{\Theta}_{\mathcal{G}_u}, \tag{4.13}$$

$$\boldsymbol{Z}_{\mathcal{G}_i}(t) = \tilde{D}_{\mathcal{G}_i}^{-1/2}\tilde{W}_{\mathcal{G}_i}\tilde{D}_{\mathcal{G}_i}^{-1/2}\boldsymbol{I}(t)\boldsymbol{\Theta}_{\mathcal{G}_i}, \tag{4.14}$$

where $\boldsymbol{Y}_{\mathcal{G}_u}(t)$ and $\boldsymbol{Z}_{\mathcal{G}_i}(t)$ are the convolved signlas for all the nodes in graph $\mathcal{G}_u$ and $\mathcal{G}_i$. Denote $\mathcal{G} = \{\mathcal{G}_u, \mathcal{G}_i\}$, $\tilde{D}_{\mathcal{G}}$ and $\tilde{W}_{\mathcal{G}}$ are the rescaled degree matrix and adjacency matrix, $\boldsymbol{\Theta}_{\mathcal{G}} \in$

$\mathbb{R}^{c\times d}$ is the graph convolutional parameters, $\boldsymbol{U}(t) = [\boldsymbol{u}_1(t), ..., \boldsymbol{u}_m(t)]^\top \in \mathbb{R}^{m\times c}$, and $\boldsymbol{I}(t) = [\boldsymbol{i}_1(t), ..., \boldsymbol{i}_n(t)]^\top \in \mathbb{R}^{n\times c}$. Note the output size d of convolutional layers may be different with the input feature size c depending on the choices of filters.

Thus, based on the factorized intensity function eq. (5.1), the intensity functions for all user-item pairs are defined as the following using a Matrix notation:

$$\begin{aligned}\boldsymbol{\Lambda}(t) &= \boldsymbol{H} + \boldsymbol{Y}_{\mathcal{G}_u}(t)\boldsymbol{Z}_{\mathcal{G}_i}(t)^\top \\ &= \boldsymbol{H} + \tilde{D}_{\mathcal{G}_u}^{-1/2}\tilde{W}_{\mathcal{G}_u}\tilde{D}_{\mathcal{G}_u}^{-1/2}\boldsymbol{U}(t)\boldsymbol{\Theta}_{\mathcal{G}_u}\boldsymbol{\Theta}_{\mathcal{G}_i}^\top[\tilde{D}_{\mathcal{G}_i}^{-1/2}\tilde{W}_{\mathcal{G}_i}\tilde{D}_{\mathcal{G}_i}^{-1/2}\boldsymbol{I}(t)]^\top, \end{aligned} \tag{4.15}$$

where $\boldsymbol{\Lambda}(t)$ is the matrix containing the intensities of all user-item pairs at time t, i.e., $(\lambda^{u,i}(t)) \in \mathbb{R}^{m\times n}$. Matrix $\boldsymbol{H} = (\eta^{u,i}) \in \mathbb{R}^{m\times n}$ is the baseline intensity matrix that captures the long-term tendency of event occurrence, $\boldsymbol{y}_u(t) \in \mathbb{R}^d$ and $\boldsymbol{z}_i(t) \in \mathbb{R}^d$ are the convolved signals for that specific user u and item i based on eq. (4.13) and eq. (4.14), where $\boldsymbol{Y}_{\mathcal{G}_u}(t) = [\boldsymbol{y}_1(t), ..., \boldsymbol{y}_m(t)]^\top \in \mathbb{R}^{m\times d}$ and $\boldsymbol{Z}_{\mathcal{G}_i}(t) = [\boldsymbol{z}_1(t), ..., \boldsymbol{z}_n(t)]^\top \in \mathbb{R}^{n\times d}$.

Specifically, the base intensity matrix $\boldsymbol{H}$ represents the long-term tendency of user u interacting with item i, which is independent of the history and the graph information. In addition, the intensity of user u interacting with item i depends on the compatibility of their instantaneous graph convoluted embedding. Because $\boldsymbol{y}_u(t)$ and $\boldsymbol{z}_i(t)$ coevolve through time and they integrate graph information by graph convolution, their inner-product measures the cumulative influence from the past interactions to the occurrence of the current event.

Incorporating the graph information enforce intensity smoothness over graphs. In addition, another benefit of the graph convolutional operation is that we reduce the dimension of the input feature drift, which is $\mathbb{R}^{|V|\times c}$, to the dimension of output feature map size $\mathbb{R}^{|V|\times d}$, where $|V|$ is the number of the vertices in graph $G = (V, \mathcal{E}, W)$.

4.3.3 Fairness Metrics

In this section, we present several new fairness metrics, which measures the inconsistency in different loss functions, such as mean square error, mean absolute error, Huber loss and Kullback-

Leibler divergence, between the intensities for different user/item groups. Without loss of generality, we assume there are s user groups (e.g., $s = 2$ for gender groups). A user's group label can be represented as a one hot vector of dimension $\mathbb{R}^s$, which indicates which group the user belongs to in all s groups. We can define the user *KL fairness* metric as:

$$\hat{U}_{KL(g||\neg g)} = \frac{1}{s}\sum_{g=1}^{s}\frac{1}{n}\sum_{j=1}^{n} E_g[\tilde{\lambda}]_j \log(\frac{E_g[\tilde{\lambda}]_j}{E_{\neg g}[\tilde{\lambda}]_j}) + \frac{1}{s}\sum_{g=1}^{s}\frac{1}{n}\sum_{j=1}^{n} E_{\neg g}[\tilde{\lambda}]_j \log(\frac{E_{\neg g}[\tilde{\lambda}]_j}{E_g[\tilde{\lambda}]_j}), \quad (4.16)$$

where $\tilde{\lambda} = \frac{1}{\lambda_{\max}}\int_0^T \lambda^{u,i}(\tau)d\tau \approx \frac{1}{\lambda_{\max}}T^{-1}\sum_{t=0}^{T}\lambda^{u,i}(t)$ is the approximated intensity integral across time and normalized by the largest intensity in training, $E_g[\tilde{\lambda}]_j$ is the average approximated intensity for the j-th item from users in group g, and $E_{\neg g}[\tilde{\lambda}]_j$ is the average approximated intensity for the j-th item from users not in group g. By applying the log sum inequality, we know that the *KL fairness* is always non-negative. When $\hat{U}_{KL(g||\neg g)} = 0$, the expected event intensities of the groups are the same. The item fairness metrics $\hat{I}$ can be defined in a similar way.

Similarly, we can present the other fairness metrics such as *MAE fairness*, *MSE fairness*, and a smoothed variation *Huber fairness* as follows:

$$\hat{U}_{MAE} = \frac{1}{s}\sum_{g=1}^{s}\frac{1}{n}\sum_{j=1}^{n}|E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j|, \quad (4.17)$$

$$\hat{U}_{MSE} = \frac{1}{s}\sum_{g=1}^{s}\frac{1}{n}\sum_{j=1}^{n}(E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j)^2, \quad (4.18)$$

$$\hat{U}_{H}=\frac{1}{2s}\sum_{g=1}^{s}\frac{1}{n}\sum_{j=1}^{n}\begin{cases}(E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j)^2, & \text{if } |E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j| \le \delta, \\ \delta|E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j| - \frac{1}{2}\delta^2, & \text{otherwise,}\end{cases} \quad (4.19)$$

where δ in eq. (4.19) is the hyperparameter. It is worth mentioning that when the difference between the average intensities within and without the group s is small, the *Huber fairness* equals the *MSE fairness*, which becomes the *MAE fairness* otherwise. They are equal at two points where $|E_g[\tilde{\lambda}]_j - E_{\neg g}[\tilde{\lambda}]_j| = \delta$. The *Huber fairness* approaches *MAE fairness* when $\delta \to 0$ and *MSE fairness* when $\delta \to \infty$ (large numbers).

In comparison with traditional non-parity fairness metrics [92, 42], our fairness metrics have two contributions. First, we consider the fairness of event prediction in a continuous time period and use the expectation of intensities over time between different groups to evaluate the fairness. Second, as the intensities are not normalized, using common fairness metrics usually lead to the explosion of the computation when the intensities are extremely large. Thus, inspired by the idea of the normalized graph Laplacian, we normalize the intensities by the largest intensity in training, which can reduce the explosion effect. It is worth mentioning that our fairness metrics all enjoy convexity and permitting efficient optimization as regularizers. We prove the convexity in theorem 3 and discuss the details of the different fairness metrics in the experiment section.

4.3.4 Maximum Likelihood Estimation with Fairness

Based on the survival analysis theory [1], the likelihood of observing a sequence of events $\mathcal{T}^{u_k,i_k} = \{t_i\}_{i=1}^n$ for a specific user-item pair (u_k, i_k) which is recorded within a time window $[0, T]$ is $\prod_{t_i \in \mathcal{T}^{u_k,i_k}} \lambda^{u_k,i_k}(t_i) \cdot \exp(-\int_0^T \lambda^{u_k,i_k}(\tau) d(\tau))$. The joint negative log-likelihood of observing all user-item pairs' events $\mathcal{O} = \{e_k = (u_k, i_k, t_k)\}_{k=1}^q$ is:

$$\mathcal{L} = -\sum_{e_k \in \mathcal{O}} \log(\lambda^{u_k,i_k}(t_i)) + \sum_{u=1}^{m} \sum_{i=1}^{n} \int_0^T \lambda^{u,i}(\tau) d\tau. \tag{4.20}$$

We further propose to incorporate fairness criteria to guide the construction of Hawkes models. Specifically, we add a penalty term to the log-likelihood function that enforces the fairness constraints. The model parameters $\boldsymbol{H}$, $\Theta_{\mathcal{G}_u}$, and $\Theta_{\mathcal{G}_i}$ are used to compute hazard rate λ in eq. (4.15) and the hazard rate is also used to compute the fairness penalty. At first glance, the objective function is non-convex due to the inner product of the two convolutional filter matrices $\{\Theta_{\mathcal{G}_u}, \Theta_{\mathcal{G}_i}\}$. Directly using common gradient based methods can usually easily trap into local minimal that result in tuning the parameters for better performance. Since our Graph Convolutional Point Process is typically one layer network, we can obtain a convex objective function by following the work [84]. Specifically, the product of two matrices can be transformed to one block matrix $\boldsymbol{\Omega} = \boldsymbol{\Theta}_{\mathcal{G}_u}\boldsymbol{\Theta}_{\mathcal{G}_i}^{\top} \in \mathbb{R}^{c \times c}$, and the model parameters can be simplified as $\boldsymbol{H}$ and $\boldsymbol{\Omega}$, which makes the likelihood eq. (4.20) convex. Furthermore, adding fairness penalties, the objective becomes:

$$OPT_f = \min_{\boldsymbol{H},\boldsymbol{\Omega}\geq 0} \mathcal{L}(\mathcal{O} \mid \boldsymbol{H}, \boldsymbol{\Omega}) + \alpha\|\boldsymbol{H}\|_* + \beta\hat{U} + \gamma\hat{I}, \tag{4.21}$$

where $\mathcal{L}$ is the log-likelihood of observed training data, $\hat{U}$ is the user fairness penalty, $\hat{I}$ is the item fairness penalty, and β and γ are the scalar trade-off parameters weighting fairness against the log-likelihood. In addition, α is a trade-off factor and $\|\cdot\|_*$ is the nuclear norm (i.e., trace norm) commonly used for matrix regularization. Since the intensities of the point processes are always non-negative, it is worth mentioning that the block matrix should also hold the non-negative constraints.

4.4 Learning and Optimization

Obviously, the non-negative constraints as well as the non-smooth nuclear norm in the objective in eq. (4.21) makes the problem hard to optimize. Following the lead of the Primal Averaging Conditional Gradient (PA-CndG) algorithm [50], which is based on Proximal Gradient (PG) [49, 66], we propose an efficient framework to optimize the objective along with the fairness penalties.

4.4.1 Alternative Formulation

In order to tackle the difficulties, we first approximate the objective function eq. (4.21) by adopting a penalty method [20, 84]. With the condition $\rho > 0$, we introduce a squared Frobenius norm of an auxiliary variable $\hat{\boldsymbol{H}}$ leading to the new following formulation:

$$\widehat{OPT_f} = \min_{\boldsymbol{H},\boldsymbol{\Omega}\geq 0,\hat{\boldsymbol{H}}} \mathcal{L}(\mathcal{O}|\boldsymbol{H}, \boldsymbol{\Omega}) + \alpha\|\hat{\boldsymbol{H}}\|_* + \rho\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|_F^2 + \beta\hat{U} + \gamma\hat{I}. \tag{4.22}$$

The new formulation eq. (4.22) divides the non-smooth nuclear norm and the non-negativity constraints, and thus benefits the optimization process. The approximate objective function is upper bounded by the real objective given ρ [20].

4.4.2 Alternating Updates

We present algorithm 5 to solve eq. (4.22) efficiently. For notation simplicity, we set:

$$f(\boldsymbol{H}, \boldsymbol{\Omega}, \hat{\boldsymbol{H}}) = \mathcal{L}(\mathcal{O}|\boldsymbol{H}, \boldsymbol{\Omega}) + \rho\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|_F^2 + \beta\hat{U} + \gamma\hat{I}. \tag{4.23}$$

We first give the following theorem and proof.

Theorem 3. *The function $f(\cdot)$ is convex and its derivative $\nabla f(\cdot)$ is Lipschitz continuous (L-smooth).*

Proof. Since the log-likelihood and the Frobenius norm are convex, we only need to prove the convexity and Lipschitz continuity of all the fairness metrics. Specifically, we prove the user fairness $U_{KL}(\cdot)$ in eq. (4.16) is convex and Lipschitz continuous gradient, and the rest of the fairness metrics can be proved similarly.

First of all, the intensity function is a linear function of parameters $\boldsymbol{H}$ and $\boldsymbol{\Omega}$ ($\boldsymbol{\Omega} = \boldsymbol{\Theta}_{\mathcal{G}_u}\boldsymbol{\Theta}_{\mathcal{G}_i}^\top \in \mathbb{R}^{c\times c}$) based on eq. (4.15) because all the other terms are features computed at a given time t. Thus, $\forall \boldsymbol{H}_1, \boldsymbol{\Omega}_1, \boldsymbol{H}_2, \boldsymbol{\Omega}_2 > 0$ and $\forall \epsilon \in (0,1)$:

$$\boldsymbol{\Lambda}(\epsilon\boldsymbol{H}_1 + (1-\epsilon)\boldsymbol{H}_2, \epsilon\boldsymbol{\Omega}_1 + (1-\epsilon)\boldsymbol{\Omega}_2) \leq \epsilon\boldsymbol{\Lambda}(\boldsymbol{H}_1, \boldsymbol{\Omega}_1) + (1-\epsilon)\boldsymbol{\Lambda}(\boldsymbol{H}_2, \boldsymbol{\Omega}_2) \tag{4.24}$$

Note that the user fairness function $U_{KL}(\cdot)$ is a function of intensities $\tilde{\lambda}$ and the log ratio $\frac{E_g[\tilde{\lambda}]_j}{E_{\neg g}[\tilde{\lambda}]_j}$ keeps the same when λ multiplies a constant. Thus we have:

$$\begin{aligned}
&U_{KL}(\epsilon\boldsymbol{H}_1 + (1-\epsilon)\boldsymbol{H}_2, \epsilon\boldsymbol{\Omega}_1 + (1-\epsilon)\boldsymbol{\Omega}_2)) \\
&\le U_{KL}(\boldsymbol{\Lambda}(\epsilon\boldsymbol{H}_1 + (1-\epsilon)\boldsymbol{H}_2, \epsilon\boldsymbol{\Omega}_1 + (1-\epsilon)\boldsymbol{\Omega}_2)) \\
&\le U_{KL}(\epsilon\boldsymbol{\Lambda}(\boldsymbol{H}_1, \boldsymbol{\Omega}_1) + (1-\epsilon)\boldsymbol{\Lambda}(\boldsymbol{H}_2, \boldsymbol{\Omega}_2)) \\
&\le \epsilon U_{KL}(\boldsymbol{\Lambda}(\boldsymbol{H}_1, \boldsymbol{\Omega}_1)) + (1-\epsilon)U_{KL}(\boldsymbol{\Lambda}(\boldsymbol{H}_2, \boldsymbol{\Omega}_2)) \\
&\le \epsilon U_{KL}(\boldsymbol{H}_1, \boldsymbol{\Omega}_1) + (1-\epsilon)U_{KL}(\boldsymbol{H}_2, \boldsymbol{\Omega}_2),
\end{aligned} \tag{4.25}$$

thus, the *KL fairness* function eq. (4.16) is convex. Since it is linear, the derivative function is constant and is exactly Lipschitz continuous. □

Similarly, the convexity property holds for both *MAE fairness* and *MSE fairness*. Also, the derivative function of *MSE fairness* satisfies the Lipschitz continuity. For the *MAE fairness*, the derivative function is the signum function, which is locally Lipschitz continuous in any interval not containing zero. Note that our *MAE fairness* ideally achieves zero and the intensities are always non-negative, we can optimize it from one side by clipping the fairness gradient to enforce non-negative. The *Huber fairness*, which is the combination of *MSE fairness* and *MAE fairness*, is continuous around zero. The convexity and Lipschitz continuity still holds for Huber.

As we know the nuclear norm term in the alternative formulation eq. (4.23) is convex, thus, based on the convexity and Lipschitz continuity proof above, we can apply cheap projected gradient descent for $\boldsymbol{H}, \boldsymbol{\Omega}$ and cheap linear minimization for $\hat{\boldsymbol{H}}$ in each iteration. Specifically, the algorithm consists of two main alternating subroutines:

• **Proximal Gradient.** Based on the schema in [66], we apply gradient update for parameters $\boldsymbol{H}$ and $\boldsymbol{\Omega}$ in each iteration in algorithm 5. Specifically, we directly calculate the proximal operator of $\boldsymbol{H}, \boldsymbol{\Omega}$ along with the constraints, which reduce to the simple projections as follows:

$$\begin{aligned}\boldsymbol{H}^k &= \underset{\boldsymbol{H}^k \geq 0}{\operatorname{argmin}}\{\frac{1}{2\xi_k}\|\boldsymbol{H}^k - (\boldsymbol{H}^{k-1} - \xi_k \nabla_1 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1}))\|^2\} \\ &= (\boldsymbol{H}^{k-1} - \xi_k \nabla_1 (f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})))_+, \end{aligned} \tag{4.26}$$

and

$$\begin{aligned}\boldsymbol{\Omega}^k &= \underset{\boldsymbol{\Omega}^k \geq 0}{\operatorname{argmin}}\{\frac{1}{2\xi_k}\|\boldsymbol{\Omega}^k - (\boldsymbol{\Omega}^{k-1} - \xi_k \nabla_2 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1}))\|^2\} \\ &= (\boldsymbol{\Omega}^{k-1} - \xi_k \nabla_2 (f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})))_+, \end{aligned} \tag{4.27}$$

where $(\cdot)_+$ simply sets the negative values to zero.

• **Conditional Gradient.** For $\hat{\boldsymbol{H}}$, we do not directly compute gradients by eq. (4.26) and eq. (4.27). We adopt a local linear expansion instead, to approximate the value, which is well known as the conditional gradient (Frank-Wolfe) method. The difference between this conditional gradient and other traditional conditional gradient methods is that the search direction, which is $\nabla_3 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})$, is defined. It is actually a variant of Nesterov's method [66] by replacing the prox-mapping with a simpler linear expansion:

$$\boldsymbol{U}^k = \arg\min_{\hat{\boldsymbol{H}}}\{\langle \nabla_3 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1}), \hat{\boldsymbol{H}}\rangle + \alpha\|\hat{\boldsymbol{H}}\|_*\}. \tag{4.28}$$

Specifically, we first calculate the top singular vector pairs of $-\nabla_3 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})$ and then followed by a line search to produce a scaling factor [93] to solve the problem.

4.5 Convergence and Complexity

4.5.1 Convergence Rate

The projected gradient method achieves the well-known optimal rate $O(1/k)$, i.e., a rate of $O(1/\epsilon)$ with the condition that learning rate $\xi_k \leq 1/L$. For the conditional gradient method, it also reaches the optimal rate $O(1/k)$ if it follows step size policy (1): $\gamma_k = \frac{2}{k+1}$ or (2): $\gamma_k =$

Algorithm 5: Graph Latent Feature Processes

```
Input: All the sequences of training events $\mathcal{O} = \{e_k = (u_k, i_k, t_k)\}_{k=1}^{q}$; learning rate $\xi_k$;
       parameters $\rho$, $\alpha$, $\beta$, $\gamma$; step size $\gamma_k \in [0, 1]$;
Output: $\boldsymbol{H}$, $\boldsymbol{\Omega}$
Choose to initialize $\boldsymbol{H}^0$, $\boldsymbol{\Omega}^0$, $\hat{\boldsymbol{H}}^0$
for $k \leftarrow 1$ to MaxIter do
    Compute the proximal operator for $\boldsymbol{H}$ and $\boldsymbol{\Omega}$:
    $\boldsymbol{H}^k = (\boldsymbol{H}^{k-1} - \xi_k \nabla_1 (f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})))_+$;
    $\boldsymbol{\Omega}^k = (\boldsymbol{\Omega}^{k-1} - \xi_k \nabla_2 (f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1})))_+$;
    Use a local linear expansion of $f$ for $\hat{\boldsymbol{H}}$:
    $\boldsymbol{U}^k = \text{argmin}_{\hat{\boldsymbol{H}}} \{\langle \nabla_3 f(\boldsymbol{H}^{k-1}, \boldsymbol{\Omega}^{k-1}, \hat{\boldsymbol{H}}^{k-1}), \hat{\boldsymbol{H}} \rangle + \alpha \|\hat{\boldsymbol{H}}\|_*\}$;
    Set $\hat{\boldsymbol{H}}^k = (1 - \gamma_k) \hat{\boldsymbol{H}}^{k-1} + \gamma_k \boldsymbol{U}^k$;
end
```

$\arg\min_{\gamma \in [0,1]} f((1-\gamma)\boldsymbol{X}^{k-1} + \gamma \boldsymbol{U}_1^k, (1-\gamma)\boldsymbol{Z}^{k-1} + \gamma \boldsymbol{U}_2^k)$ [50]. The algorithm should still reach the $O(1/k)$ optimal rate in general by properly choosing the learning rate and the step size parameter.

4.5.2 Computational Complexity and Scalability

The computational complexity is $O(N^2 E/\epsilon)$, where the constant E is the time complexity calculating the gradient of each entry in the parameter matrix, and $N = max(m, n)$ is the maximal number of users or items. Note that only parameter matrix $\boldsymbol{H}$ in the intensity function eq. (4.15) is in the space of $\mathbb{R}^{m \times n}$.

To improve computational efficiency, we can use some prior knowledge to approximate the base intensity $\eta^{u,i}$ as a fixed parameter that captures the long-term tendency of event occurrence of (u, i) pair. For example, we can count the number of events in the training period of that entry. Then the parameter space can be reduced to $\mathbb{R}^{c \times c}$, as we only have one parameter $\boldsymbol{\Omega}$ to optimize and $c \ll N$. Also, since we don't need to calculate the non-smooth nuclear norm, the objective function can be easily optimized by gradient descent, which still has the well known convergence rate $O(1/k)$ for a convex and L-smooth function, and the computational complexity for the optimization is $O(c^2 E/\epsilon)$. If the parameter space $\mathbb{R}^{c \times c}$ is still huge, we can also use the stochastic gradient descent with convergence rate $O(1/\sqrt{k})$ to minimize mini-batch for a convex and L-smooth function, and achieve complexity of $O(E/\epsilon^2)$.

Table 4.2: Dataset description.

Dataset	User	Item	Event	Pair	Sensitive Feature	Time
IPTV	7100	436	$2.4M$	4726	Genres	8040
Yelp	100	$17K$	$35K$	20246	N/A	44640
Reddit	1000	1403	$10K$	2053	N/A	4090
LinkedIn	2439	82	7495	5601	Age	40

4.6 Experiment and Results

In this section, we introduce the experiment and results.

4.6.1 Data

Our model is evaluated on four real world datasets which include user or item features as well as temporal interactions between them. The details are shown in table 4.2. Specifically, the **IPTV** dataset [87] contains 7,100 users and 436 TV programs from January to November 2012 with several program features, and some features such as genres and countries can be regarded as sensitive feature on which we will apply fairness penalty. The **Yelp**[1] dataset is presented by Yelp dataset challenge. It contains the time stamps of writing reviews for 17k businesses by 100 users in 11 years. The **Reddit**[2] dataset records the time stamps of posting discussions between random selected 1,000 users and 1,403 threads in January 2014. The **LinkedIn** dataset [88] contains the job hopping records between 2,439 LinkedIn users and 82 IT companies. We group the users by old and young as the fairness sensitive features. We compare the state-of-art baselines with our model using the first three datasets, and evaluate the fairness on the sensitive features of **IPTV** and **LinkedIn** datasets.

Following [65], we construct a user or item graph as an unweighted k-nearest neighbor graph in the space of features such as TV features. If user and item features are not available, we can apply SVD to a two-dimensional user-item matrix, which is constructed from the time sequences and each entry represents the total number of user-item interactions, to get latent feature vectors for the user and item. We split all of the events on the total time T by a proportion, e.g., $p = 0.76$, so the events before time $T \cdot p$ are the training data, and the rest of them are testing data.

[1]https://www.yelp.com/dataset/challenge

[2]https://dynamics.cs.washington.edu/data.html

4.6.2 Evaluation Metrics

We apply three metrics to evaluate the performance and the fairness of the model:

Item Relevance: For a specific user u with the event history $\mathcal{T} = \{t_i\}_{i=1}^n$, we can calculate the survival of all the items at each testing time t, which is $S_i(t) = \exp(-\int_{t_n^i}^{t} \lambda_i(\tau)d(\tau))$. After ordering all the survivals, we can obtain the rank of the ground-truth item the user actually interacts with at testing time t. Therefore, the ground-truth item should be ranked one ideally. We report the *mean average rank* (MAR) of all testing cases following [84]. A smaller MAR value indicates better prediction performance.

Time Prediction Accuracy: For a specific user-item pair (u, i), we record the *mean absolute error* (MAE) between the next predicted time and the ground truth testing time t. We compute the predicted time by the density of next event time as $f(t) = \lambda_{(u,i)}(t)S_{(u,i)}(t)$, and then predict the next event by expectation. Besides, we also present the relative percentage of the time prediction error (Err %).

Fairness Loss: we evaluate the fairness loss defined in Section 4.3.3 on the test set. The test set loss may change due to the trade-off parameters in minimizing the objective function using training data. It measures the degree of fairness. A larger fairness loss indicates less fairness of the model.

4.6.3 Baseline Methods

Poisson: The Poisson process is the relaxation of our model without considering history dependencies. It only has one constant parameter, the base intensity η.

Poisson-Tensor: Poisson-Tensor uses Poisson regression error rather than RMSE as the tensor decomposition loss function [14]. The entries in the tensor are the numbers of events in each discretized time slot, and it assumes that the missing values obey Poisson distribution, which is more reasonable, instead of Gaussian. The intensity of the test data can be simulated by using either the intensity we get in the last time interval, or the average of the intensities over all the training intervals. We present the best performance of them.

LR-Hawkes: Low-rank Hawkes is a collection of Hawkes processes [20], which assumes that each process is independent with others and the parameters matrices have low rank structure. Neither the correlations by interaction nor the dependencies between users or items are considered.

Coevol: Coevolve is a coevolutionary latent feature process [84] which captures the coevolutionary dynamics of user and item features. It enhances the prediction performance by adding coevolutional interaction features with base features. However, it only considers interaction events while ignoring correlations between user-user pairs or item-item pairs in the real world scenario.

GCN-Hawkes: GCN-Hawkes learns the Hawkes process parameters under a graph convolutional recurrent network [75]. It is worth mentioning that our proposed work is on a different direction with GCN-Hawkes. GCN-Hawkes actually adopts "GCN layer and LSTM pooling" network to learn the Hawkes coefficients (e.g., base intensity and influence scalar). The graph feature embedding is static and does not consider the coevolutionary dynamics. However, our model directly integrates the graph information to user and item features by graph convolutional embedding.

4.6.4 Comparison with Baselines

We compare our model with some state-of-art baselines in table 4.3 by evaluating the item relevance prediction and the returning time prediction in IPTV, Yelp and Reddit dataset. We can see that in general, our model outperforms other methods. Our model and the *GCN-Hawkes* are much better than other methods due to the graph convolution feature embedding, especially for the data sets with fewer temporal events, e.g., Yelp and Reddit. The significant improvement of these geometric related models demonstrates that the additional correlations between the users or items are efficiently extracted and they are helpful for the model prediction performance. Also, as our model and the *Coevol* adopt coevolutionary dynamics, the returning time prediction becomes more accurate. The coevolutionary dynamics can efficiently utilize the history events, which is important for returning time prediction. Besides, the history of the events is crucial in predicting future events. Poisson process based methods (e.g., *Poisson* and *Poisson-Tensor*), which drop much of the history information, perform worse comparing with others.

Table 4.3: Average prediction performance comparison on IPTV, Yelp, and Reddit datasets.

Datasets	Metrics	Methods					
		Ours	GCN-Hawkes	LR-Hawkes	Coevol	Poisson	Poisson-Tensor
IPTV	MAR	4.757	**1.643**	5.175	13.57	173.7	178.7
	MAE	**45.52**	361.0	822.1	160.3	993.1	933.6
	Err %	**0.7895**	5.13	12.27	2.35	14.83	13.89
Yelp	MAR	**88.03**	94.62	116.0	671.2	7778	1738
	MAE	**398.3**	499.0	845.7	587.3	850.9	587.1
	Err %	**11.84**	14.59	23.71	17.49	23.91	17.48
Reddit	MAR	**3.114**	6.010	49.14	82.44	128.2	85.49
	MAE	**2126**	5367	8476	5323	10314	9155
	Err %	**2.73**	14.15	21.50	14.27	26.59	24.09

Table 4.4: Average prediction performance comparison of different propagation models on IPTV dataset.

Description	Propagation Model	Evaluation Methods		
		MAR	MAE	Err %
$K = 2$ Chebyshev filter eq. (4.5)	$\boldsymbol{Z} = \sum_{k=0}^{K} T_k(\tilde{L})\boldsymbol{X}\boldsymbol{\Theta}_k$	22.51	70.80	3.8
$K = 3$ Chebyshev filter eq. (4.5)		33.52	71.66	3.8
Single parameter 1^{st}-order model	$\boldsymbol{Z} = (I_n - D^{-1/2}WD^{-1/2})\boldsymbol{X}\boldsymbol{\Theta}$	10.72	69.62	3.6
Opposite parameter 1^{st}-order model	$\boldsymbol{Z} = (I_n + D^{-1/2}WD^{-1/2})\boldsymbol{X}\boldsymbol{\Theta}$	17.43	55.50	1.7
Renormalization trick eq. (4.7)	$\boldsymbol{Z} = \tilde{D}^{-1/2}\tilde{W}\tilde{D}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta}$	**4.757**	**45.52**	**0.79**
1^{st}-order term only	$\boldsymbol{Z} = D^{-1/2}WD^{-1/2}\boldsymbol{X}\boldsymbol{\Theta}$	12.88	65.76	3.2
Multi-layer perception	$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{\Theta}$	13.57	160.3	2.4

4.6.5 Comparison with Different Graph Propagation Models

We compare the prediction performance with different graph propagation models [44] on the IPTV dataset. The results are shown in table 4.4. In general, these propagation models achieve comparable performance with other baselines we present before, and our model, the *renormalization trick*, outperforms the others. The *Chebyshev filter* model exhibits overfitting with the increase of K, since it contains more parameters, and our model is a linear approximate variation with $K = 1$. As our datasets are not very big, using simplified version of graph propagation model such as several first-order models is enough to achieve good performance. Also, when using *multi-layer perception*, it reduces to *Coevol* model while not applying the interaction features.

Table 4.5: The fairness performance of different metrics on IPTV dataset.

Fairness Loss in Objective	MAE	MSE	Huber	KL
MAE Evaluation	0.1760	0.1832	0.1749	**0.1706**
MSE Evaluation	0.0324	0.0336	0.0306	**0.0291**
Huber Evaluation	0.1783	0.1822	0.1824	**0.1683**
KL Evaluation	0.1006	0.1002	**0.0966**	0.1002
Average Evaluation	0.1218	0.1248	0.1211	**0.1171**

4.6.6 Comparison of Different Fairness Metrics

We investigate the effect of using different fairness metrics in a way similar to [92]. Specifically, we use each fairness metric as a regularizer in the objective function with the same trade-off parameter and train the model. After a fixed number of iterations, we evaluate the model performance using all four fairness metrics. The results are shown in table 4.5. Each column indicates the type of fairness loss metric used in the objective for training the model and each row indicates the type of fairness loss used in evaluation. For example, if we use *MAE fairness* to train a model and *MSE fairness* to evaluate the model, the fairness loss is 0.0324. The last row is the average number of each column. A smaller fairness loss indicates a more fair scenario.

At first, It it obvious that our learning method can efficiently optimize the objective and present stable results under different fairness loss metrics. Second, we can see that after using *KL fairness* in the optimization, we obtain the smallest fairness loss under most of the metrics, which indicates that the *KL fairness* are more efficient and less costly in the optimization. Further, even though we use the normalized intensity for computing the fairness loss, the value of *MSE fairness* evaluation is always smaller than others. The reason is that it's quadratic and the input intensities are smaller than one, which lead to a much smaller number. Thus, when using *MSE fairness*, we need to fine tune the trade-off parameters to control the balance between precision and fairness.

4.6.7 Trade-off between Fairness and Accuracy Measures

We follow [5] to investigate the trade-off between accuracy and fairness for two applications: IPTV item parity and LinkedIn user parity. In IPTV dataset, we treat 22 genre groups as the sensitive feature. In LinkedIn dataset, we treat user age as the sensitive feature. The users are

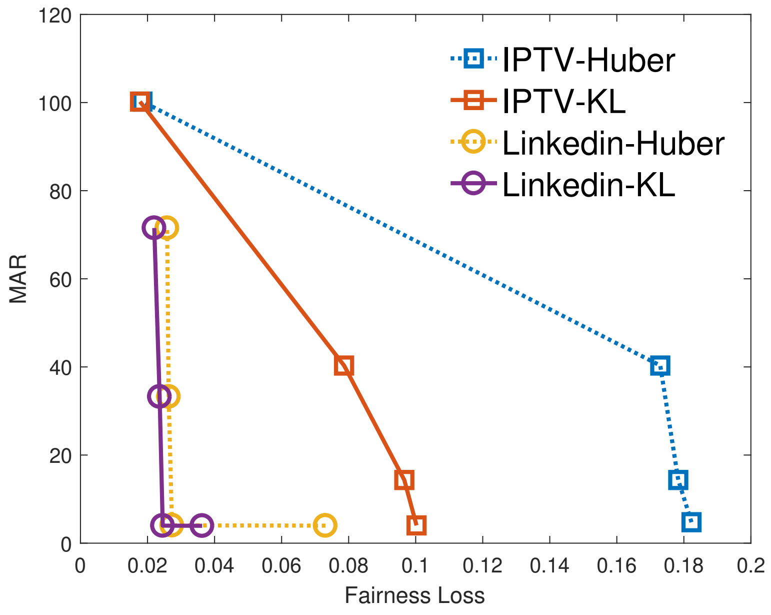


Figure 4.1: Trade-off between prediction and fairness of IPTV and LinkedIn datasets.

divided into young and old, with a 64% and 36% split. The trade-off parameter of the fairness β or γ varies in the range $[0, 1, 100, 10000]$ in the training stage. For each value of trade-off parameter, we obtain a model which minimizes the associated regularized loss function. As the *Huber fairness* incorporates the *MAE fairness* and *MSE fairness*, we only demonstrate the *Huber fairness* and the *KL fairness* on IPTV dataset for item fairness and on LinkedIn dataset for user fairness. The evaluation metric of fairness loss on test set is the same as the one used in the regularizer.

The relations between test set fairness and accuracy are shown in fig. 4.1. In the figure, y axis is MAR representing the model prediction accuracy, and x axis is the fairness loss on test set. The model becomes more fair when the fairness loss is smaller and ideally approaches 0. The prediction performance is better when the MAR is smaller and ideally approaches 1. There are four curves in total each corresponding to the model frontier using one of the two fair metric regularizers on one of the two data sets. Four markers on each curve, approximately from left to right, indicate the models trained using different trade-off parameters changing from $[10000, 100, 1, 0]$. Generally, as the trade-off parameter decreases, the model becomes more accurate and less fair.

In addition, fig. 4.1 shows the great diversity of tradeoffs across different datasets and different fairness regularizers. For example, if we examine the *KL fairness* and *Huber fairness* on IPTV, the curvature is relatively mild, which indicates an approximately fixed rate at which fairness can be traded for accuracy. In contrast, on dataset LinkedIn, fairness loss can be reduced to some small threshold value without decreasing event prediction accuracy dramatically, which indicates a relatively easy gain of fairness. It is worth mentioning that the *KL fairness* is usually smaller than the *Huber fairness* when the same trade-off parameter is chosen, which is consistent with our experiment in table 4.5.

CHAPTER 5
LIST-WISE FAIRNESS CRITERION FOR POINT PROCESSES

5.1 Introduction

Many types of event sequence data exhibit triggering and clustering properties in space and time. For example, after a large earthquake, events of after-shocks usually occur in the following days or weeks near the epicenter of the main shock [28]. Similarly, criminologists have reported that 25% to 50% crime events are observed in a few areas of a city [61]. They have also demonstrated that certain types of crime events such as burglaries are often reported repetitively from the same neighborhood [6]. The time interval and spatial distance between two events carry a great deal of information about the underlying dynamics of a specific type of events.

Predicting and ranking the rate of events as a function of space and time enables important applications. Typically, space is divided into regions, time is divided into short intervals, and regions are ranked based on the predicted event rates over a time window. For example, in a predictive policing system, a city is divided into geographic sub-regions such as grid cells or political boundaries. A predictive algorithm is used to forecast the rates of crime events for each region at each day based on historical crime events. According to the predicted rates, police daily patrol activities can be adjusted so that more resources are allocated to the regions with higher risks. In practice, due to limited resources, regions are ranked by the predicted hazard rates in each day and police activities are directed to top-k regions, also known as hotspots [61].

Variations of point-process models have become very popular for modeling event rates based on historic events. Most of them assume certain forms of dependency on the history in different ways. For example, the Hawkes process [27, 55] assumes that the influences from past events are linearly additive towards the current event. Such models can capture inhomogeneous inter-event times and causal (temporal) correlations observed in events. Spatial-temporal Hawkes models extend temporal models to predict the rate of events at a specific location and time. Spatial heterogeneity in hazard rates can be characterized as base intensities and the self-exciting effects can be modeled with a variety of temporal kernels. Model parameters can be estimated using standard maximum likelihood estimators given training data, e.g., events observed before a specific time.

Although those predictive models improve event forecasting accuracy, biased predictions may be introduced and amplified due to factors such as data bias and the feedback loop of algorithms. For example, time-stamped geo-tagged event data from Twitter have been used for rapid flood mapping, damage assessment, and situation awareness [11]. However, it has been reported that higher disaster-related Twitter-use communities tend to be of higher socioeconomic status [98]. Prediction based on such data may exhibit socioeconomic bias. Moreover, recent studies have focused on the bias problem of event prediction in predictive policing. One potential problem is that if the police only patrol areas with higher estimated risks, there will likely be more arrests than in other areas, and then biased arrests may be further amplified through the feedback loop.

While there have been some early explorations [90, 46, 95] in developing ranking fairness metrics that can be adopted by hazardous event prediction, most of them focus on either measurement of fairness or post-processing ranking list to satisfy a fair condition. Hence, the ranking functions are not influenced by the fairness metrics. A recent work [62] introduces demographic parity into spatial-temporal crime prediction and directly uses it to penalize the likelihood function. The fairness metric enforces the amount of police patrol allocated to each demographic group in selected hotspots to be proportional to the percentage of that group in the whole population. However, the fairness metric does not guarantee group parity at any point in ranked regions.

In this chapter, we propose a novel list-wise fairness criterion for spatial-temporal point process, which can efficiently evaluate the list ranking fairness. We also present a strict definition of the unfairness consistency property of a fairness metric and prove that our list-wise fairness criterion satisfies this property. We further integrate the fairness criterion into the objective function and then obtain a fairness-aware ranking function that can generate a fair ranking list. We carry on experiments over several real-word spatial-temporal datasets, and the results demonstrate the effectiveness of our list-wise fairness criterion. We also discuss the scalability of our method and propose a smoothed variation, which makes it easier for optimization.

5.2 Motivation

In this section, we first introduce the background of spatial-temporal point process model for event prediction and then present the fairness concerns on event prediction.

5.2.1 Ranking Prediction by Spatial-Temporal Point Process

A collection of n events in an area (e.g., a city) during a time window $[0, T]$ is represented as a temporally ordered list $\mathcal{T} = \{e_k = (x_k, t_k)\}_{k=1}^{n}$, where event e_k happens at time t_k and location x_k, e.g., a pair of longitude and latitude. An event e_k may be a crime event reported from a victim or a disaster-related rescue request. In many event prediction applications, an area is divided into grid cells or political boundaries such as ZIP Codes. For example, disaster areas can be discretized into 30m by 30m square grid cells (resolution of TM remote sensing image), 150m by 150m (size of a city block), or larger. Let $\mathcal{G}$ denote the set of grid cells and $g \in \{1, 2, \dots, m\}$ index all m cells in $\mathcal{G}$. For each location x_k, let g_k denote the index of the cell that covers this location.

A temporal event sequence at the g^{th} grid cell can be modeled as a Hawkes process [60, 62]. The process can be characterized via its conditional intensity $\lambda_g(t)$, which models the expected rate of the event occurrences at the cell given the history of all the previous events up to time t. The conditional intensity function is:

$$\lambda_g(t) = \eta_g + \sum_{t_k < t, g_k = g} \theta \kappa_\omega(t - t_k), \tag{5.1}$$

where $\eta_g > 0$ is the base intensity of cell g, θ is the self-exciting coefficient, and κ_ω is the kernel function that captures the temporal intensity triggered by recent events. A common choice of kernel functions is an exponential kernel function with a bandwidth ω, i.e., $\kappa_\omega(t) = \omega \exp(-\omega t)$.

The base intensity η_g can be modeled as a function of spatial covariates/features, such as demographics [80, 62], geological and socioeconomic variables. Let $\boldsymbol{f}_g$ denote the d-dimensional feature vector for cell g, i.e., $\boldsymbol{f}_g \in \mathbb{R}^d$. Commonly, the base intensity is log-linear with the coefficients $\boldsymbol{\alpha}$ and the feature vector $\boldsymbol{f}_g$, that is, $\eta_g = \exp(\boldsymbol{\alpha} \cdot \boldsymbol{f}_g)$. The base intensity can be inhomogeneous through the space, which explains spatial variations of event hazards (e.g., disparate crime rates or flood hazards in different neighborhoods).

Given the observed historic event sequences $\mathcal{T}$, the model parameters can be estimated by maximizing the log-likelihood [62], or equivalently, minimizing the joint negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\alpha}, \theta, \omega) = -\sum_{k=1}^{n} \log(\lambda_{g_k}(t_k)) + \sum_{g \in \mathcal{G}} \int_0^T \lambda_g(t) dt, \tag{5.2}$$

where g_k is the cell index for event e_k.

To better capture correlations between multiple processes defined on different grid cells, we can incorporate spatial proximities in the form of graphs into Hawkes processes. Specifically, each grid cell is a node and two nodes are connected by an edge if they are neighborhoods. The graph is a proximity network of spatial cells. Similar to [41, 69, 75], a graph regularization is added to enforce spatial smoothness of the intensities at each cell. Formally, the objective function of the spatial-temporal Hawkes process with graph regularization is:

$$\mathcal{O}(\boldsymbol{\alpha}, \theta, \omega) = \min \mathcal{L}(\boldsymbol{\alpha}, \theta, \omega) + \rho\{T^{-1} \sum_{t=1}^{T} tr(\boldsymbol{\Lambda}(t)^\top L \boldsymbol{\Lambda}(t))\}, \tag{5.3}$$

where $\boldsymbol{\Lambda}(t) = [\lambda_1(t), \lambda_2(t), ..., \lambda_m(t)]^\top$ is the vector of event rates at m cells during a time window, ρ is the regularization parameter, and L is the Laplacian matrix constructed on the graph.

For event forecasting, the model parameters estimated using training data can be used to compute the intensities in eq. (5.1) at each cell g and a given time t. A higher intensity means a larger probability that an event will happen at its corresponding location. In practice, the intensities $\boldsymbol{\Lambda}(t)$ in the list are ranked from the highest to the lowest, and top-K hotspots may be selected at time t for informing further activities such as police patrolling.

Table 5.1: Example for ranking fairness.

Cell by Rank	Intensity	Group	Race 1	Race 2
Cell 1	10.0	1	10.0	1.0
Cell 2	5.0	2	5.0	6.0
Cell 3	4.6	2	5.0	6.0
Cell 4	4.2	2	4.0	7.0
Cell 5	3.8	2	3.0	8.0
Cell 6	3.4	2	2.0	9.0
Cell 7	3.0	2	1.0	10.0
Cell 8	2.8	1	9.0	2.0
Cell 9	2.4	1	8.0	3.0
Cell 10	0.8	1	9.0	4.0

5.2.2 Fairness Concerns on Ranking

Our concern is that grid cells ranked by event hazard rates can make the visibility of a disadvantaged group even worse. For instance, the disadvantaged groups can be racial minorities or the communities of lower social economic status. Existing fairness metrics focus on the group fairness averaged over the entire list such that the average amount of attention received by each demographic group should be fair. However, they do not compare the group fairness at every point in the ranked list.

In table 5.1, we list a simple example to demonstrate that event rate prediction may exhibit bias towards certain groups. Assume there are 10 locations (e.g., grid cells), which are ranked by the predicted event rates during a given time period. Each cell is associated with 2-dimensional demographic feature and each feature indicates the population of one race in the cell. The column “Group” indicates the type of majority race for each cell. For example, 1 means that race 1 is the majority and 2 means race 2 is the majority in that cell. Specifically, for cell 1 in the first row, the predicted hazard rate is 10.0, which is the highest. There are 10.0 persons of race 1 and 1.0 person of race 2 living in the area of cell 1.

If we use the traditional fairness metrics to evaluate the entire list in table 5.1, it is fair. Specifically, we can see that the total numbers of population for race 1 and race 2 are the same, which is 56.0. Also, there are six cells labeled as group 2 while four labeled as group 1. However, it is not fair at every point of the list. As we can see, most of the locations on the top of the list are labeled as group 2, which means the ranker intends to rank group 2 higher than group 1. Moreover, in reality, only top ranked cells receive sufficient attentions. If we take top-5 locations into consideration, there are 80% of the locations labeled as group 2, which is also unfair for race 1. In this case, a more specific fairness criterion focusing on the entire ranking list is needed.

5.3 List-wise Fairness Criterion

In this section, we introduce our **List-wise Fairness Criterion** for spatial-temporal point process. We first propose a series of definitions to describe the **unfairness consistency** property of a fairness metric and then prove that our metric satisfies this property.

5.3.1 Preliminaries

Let $\mathcal{G}$ be the instance space, e.g., a set of all m grid cells. Each instance is associated with some sensitive features such as races or social economic status. For simplicity, we assume there are two sensitive features, such as race type one and race type two. Let $\mathcal{Y}$ be the set of values for one race for all grid cells, and $\tilde{\mathcal{Y}}$ be the set of values for the other, respectively. The feature value of each instance indicates the relevance of the instance with respect to that feature. For example, if we define the feature as the race population in a cell, $y = 10.0$ means 10.0 population of race type one and $y = 0$ means zero population of that type. Also, a larger $y \in \mathcal{Y}$ indicates a larger representation of race type one.

At a specific time t, the intensity function $\lambda_g(t)$ can be considered as a mapping from instances $\mathcal{G}$ to $\mathbb{R}$ and be shortened as λ_g. For every instance $g \in \mathcal{G}$, we rank them by intensity function λ_g. The final ranking list is denoted by $g_{(1)}, ..., g_{(m)}$, which satisfies $\lambda_{g_{(1)}} \geq ... \geq \lambda_{g_{(m)}}$. Let $y_1, ..., y_m (y_i \in \mathcal{Y})$ and $\tilde{y}_1, ..., \tilde{y}_m (\tilde{y}_i \in \tilde{\mathcal{Y}})$ be the sensitive features associated with $g_1, ..., g_m$, respectively. Denote $S_m = \{(g_1, y_1, \tilde{y}_1), ..., (g_m, y_m, \tilde{y}_m)\}$ the set of intensities and features. Following [29, 16, 85], we assume that $(g_i, y_i, \tilde{y}_i)$ are i.i.d. samples drawn from an underlying distribution $P_{GY\tilde{Y}}$ over $\mathcal{G} \times \mathcal{Y} \times \tilde{\mathcal{Y}}$.

The Normalized Discounted Cumulative Gain (NDCG) is a widely used list-wise ranking metric to measure the ranking quality. It is often used to measure if web search engine algorithms rank most relevant documents at top ranks. In our case, we adopt its formula to measure the relevance of a ranked list with respect to sensitive feature values. Thus, we replace the relevance scores with the sensitive feature values in the following definition.

Definition 4. *Let $P(r)(r \leq 1)$ be a discount function on ranking positions. The intensity function λ_g is the ranker. The Discounted Cumulative Gain (DCG) of the ranker λ_g with respect to a sensitive feature $\mathcal{Y}$ using a discount function $P(r)$ is defined as:*

$$DCG(\lambda_g, \mathcal{G}, \mathcal{Y}) = \sum_{r=1}^{m} y_{(r)} P(r). \tag{5.4}$$

We can similarly define DCG for another sensitive feature $\tilde{\mathcal{Y}}$ as $DCG(\lambda_g, \mathcal{G}, \tilde{\mathcal{Y}}) = \sum_{r=1}^{m} \tilde{y}_{(r)} P(r)$.

The ideal DCG (IDCG) is the best DCG value of any possible ranking function with respect to a sensitive feature. Specifically, for the sensitive group $\mathcal{Y}$, we have $IDCG(\mathcal{G}, \mathcal{Y}) = \max_{\lambda_g'} \sum_{r=1}^{m} y'_{(r)} P(r)$. Thus, the Normalized DCG of intensity function λ_g on S_m with discount function $P(r)$ is defined as:

$$NDCG(\lambda_g, \mathcal{G}, \mathcal{Y}) = \frac{DCG(\lambda_g, \mathcal{G}, \mathcal{Y})}{IDCG(\mathcal{G}, \mathcal{Y})}. \tag{5.5}$$

NDCG are normalized scores ranging from 0.0 to 1.0 and thus are cross-group comparable. A NDCG is a standard NDCG if the discounting function is chosen to be the inverse logarithm decay $P(r) = \frac{1}{\log(r+1)}$. The choice of the base of the logarithm does not affect NDCG since the normalization can cancel out constant scaling. We use the natural logarithm in this chapter. It is worth mentioning that even though the discount function $P(r)$ is defined on positive integers r, we treat it as a function of non-negative real variable in the following sections. Thus, we can also consider the corresponding derivative $P'(r)$ and integral $\int P(r)dr$. In the following section, we leave out the word ”standard” and directly use NDCG unless we emphasis the difference.

5.3.2 List-wise Fairness Criterion

We now propose our **List-wise Fairness Criterion** of ranked list with respect to sensitive features. Intuitively, we can compare the difference of NDCG scores with respect to different sensitive features (e.g., racial groups). A disparity between NDCG scores indicates a larger degree of unfairness between the racial groups. A strict definition of our **List-wise Fairness Criterion** between each pair of groups is:

$$F(\lambda_g, \mathcal{G}, \mathcal{Y}, \tilde{\mathcal{Y}}) = (NDCG(\lambda_g, \mathcal{G}, \mathcal{Y}) - NDCG(\lambda_g, \mathcal{G}, \tilde{\mathcal{Y}}))^2. \tag{5.6}$$

Note that an ideal **List-wise Fairness Criterion** should substantially distinguish the ranking gain with respect to two groups at any prefix of the ranking. Below we first give the formal definition that a ranker measured by a metric $\mathcal{F}$ is **consistently unfair** between two groups. The definition describe the **unfairness consistency** property of a fairness measure.

Definition 5. *Let $(g_1, y_1, \tilde{y}_1), (g_2, y_2, \tilde{y}_2), ...$ be i.i.d. instance-label pairs drawn from the underlying distribution $P_{GY\tilde{Y}}$ over $\mathcal{G} \times \mathcal{Y} \times \tilde{\mathcal{Y}}$. Given $S_m = \{(g_1, y_1, \tilde{y}_1), ..., (g_m, y_m, \tilde{y}_m)\}$ and intensity function λ_g as the ranker. The ranker λ_g measured by a fairness metric $\mathcal{F}$ is said to be* ***consistently unfair*** *between two groups if there exists a negligible function* [1] *$\mu(N)$ such that for every sufficient large N, with probability $1 - \mu(N)$,*

$$\mathcal{F}(\lambda_g, \mathcal{G}, \mathcal{Y}, \tilde{\mathcal{Y}}) > 0, \tag{5.7}$$

holds for all $m \geq N$ simultaneously.

This definition indicates the **unfairness consistency** property by a metric measuring the rank list. We then give a theorem to show that our fairness metric indeed satisfies this property. For simplicity, here we state the theorem for the simple case of features with binary values, i.e.,

[1] A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is negligible iff $\forall c \in \mathbb{N}$, $\exists n_0 \in \mathbb{N}$ such that $\forall n \geq n_0$, $\mu(n) < n^{-c}$.

$\mathcal{Y} = \{0, 1\}$ and $\tilde{\mathcal{Y}} = \{0, 1\}$. It is easy to extend the result to the general case where values of $\mathcal{Y}$ and $\tilde{\mathcal{Y}}$ are finite sets [85].

To begin with, suppose there exist another intensity function $\hat{\lambda}_g$ that preserves the order [2] as original intensity function λ_g, then we have NDCG$(\lambda_g, \mathcal{G}, \mathcal{Y})$ = NDCG$(\hat{\lambda}_g, \mathcal{G}, \mathcal{Y})$ by definition. Hence, the NDCG is defined on an equivalent class of intensity functions which can preserve the same order. We now introduce the concept of canonical form.

Definition 6. *Given an intensity function λ_g, we present a canonical form of λ_g as:*

$$\hat{\lambda}_g = \Pr_{G \sim P_G}[\lambda_G \leq \lambda_g]. \tag{5.8}$$

The benefit of using the canonical form intensity function is that it satisfies the following property, which can be easily proved by definition.

Proposition 7. *For any intensity function λ_g, its canonical form $\hat{\lambda}_g$ preserves the order of λ_g and has uniform distribution on interval $[0, 1]$.*

Now we give the following theorem:

Theorem 8. *Given the canonical intensity function $\hat{\lambda}_g$, let $y'(s) = \Pr_{G \sim P_G}[Y = 1 \mid \hat{\lambda}_G = s]$ and $\tilde{y}'(s) = \Pr_{G \sim P_G}[\tilde{Y} = 1 \mid \hat{\lambda}_G = s]$. Assume $y'(s)$ and $\tilde{y}'(s)$ are Hölder continuous in s. Then, unless $y'(s) = \tilde{y}'$ almost everywhere on interval $[0, 1]$, the ranker λ_g measured by our* ***List-wise Fairness Criterion*** *is* ***consistently unfair*** *between the groups with sensitive features $\mathcal{Y}$ and $\tilde{\mathcal{Y}}$.*

Proof. We prove our **unfairness consistency** in theorem 8 by adopting the technology provided by [85] which are used to prove the property that a measure can distinguish ranking functions. We first define the pseudo expectation $\mathcal{N}(m)$ and $\tilde{\mathcal{N}}(m)$, which are integrals to approximate the DCG, for the sensitive features $\mathcal{Y}$ and $\tilde{\mathcal{Y}}$ respectively. We start with $\mathcal{Y}$:

[2]Preserving the order means for $\forall g_1, g_2 \in \mathcal{G}, \lambda_{g_1} \geq \lambda_{g_2}$ implies $\hat{\lambda}_{g_1} \geq \hat{\lambda}_{g_2}$.

Definition 9. *Assume $\mathcal{Y} = \{0, 1\}$, and let $y'(s) = \Pr_{G \sim P_G}[Y = 1 \mid \hat{\lambda}_G = s]$, we define the pseudo expectation $\mathcal{N}(m)$ for the unnormalized DCG as:*

$$\mathcal{N}(m) = \int_1^m y'(1 - r/m)P(r)dr = m \int_{1/m}^1 y'(1 - s)P(ms)ds, \tag{5.9}$$

with the substitution of integration $r = ms$. Suppose that $F(x) = \int_1^x P(r)dr$ and the probability $p = \Pr[Y = 1] > 0$, we have the normalized pseudo expectation $\mathcal{E}(m)$ as $\mathcal{E}(m) = \mathcal{N}(m)/F(mp)$.

We first prove that the difference between the NDCG and its pseudo expectation is relatively small with high probability by theorem 10.

Lemma 10. *Suppose $p = \Pr[Y = 1] > 0$ and $y'(s) = \Pr_{G \sim P_G}[Y = 1 \mid \hat{\lambda}_G = s]$ is Hölder continuous* [3] *with constants $a, C > 0$ in $s \in [0, 1]$. Then*

$$\Pr[|\mathit{NDCG}(\lambda_g, \mathcal{G}, \mathcal{Y}) - \mathcal{E}(m)| \geq 5Cp^{-1}m^{-\min(a/3,1)}] \leq O(e^{-m^{1/4}}). \tag{5.10}$$

We then prove that the difference between the pseudo expectations for the NDCG of the two groups is much larger by theorem 11.

Lemma 11. *Suppose $p = \Pr[Y = 1] > 0$ and let $y'(s) = \Pr_{G \sim P_G}[Y = 1 \mid \hat{\lambda}_G = s]$ and $\tilde{y}'(s) = \Pr_{G \sim P_G}[\tilde{Y} = 1 \mid \hat{\lambda}_G = s]$. Then, unless $y'(s) = \tilde{y}'$ almost everywhere on interval $[0, 1]$, there must exist an integer $K \geq 0$ and a constant $B \neq 0$, so that*

$$|\mathcal{E}(m) - \tilde{\mathcal{E}}(m) - \frac{B}{\log^K m}| \leq O(\frac{1}{\log^{K+1} m}). \tag{5.11}$$

The proofs of the two lemmas are in section 5.3.3. Thus, from theorem 10 and theorem 11, and with the observation that $\sum_{m>N} e^{-m^{1/4}} \leq O(N^{3/4}e^{-N^{1/4}}) \leq O(e^{-N^{1/5}})$, the ranker λ_g measured by our **List-wise Fairness Criterion** is **consistently unfair** between two groups with high probability. $\square$

[3] That is, for $\forall s, s' \in [0, 1], |y'(s) - y'(s')| \leq C\|s - s'\|^a$

5.3.3 Proofs of the Lemmas

In this section, we present the proofs of theorem 10 and theorem 11. We first give two key claims that are useful to prove theorem 10, and then provide the proof of theorem 11. The proofs of claims are in section 5.3.4.

Proof of Lemma 7

We first give two key claims.

Claim 12. *Suppose that* $F(x) = \int_1^x P(r)dr$ *and the probability* $p = \Pr[Y = 1] > 0$. *For every sufficiently large* m, *the following inequality*

$$|NDCG(\lambda_g, \mathcal{G}, \mathcal{Y}) - \frac{DCG(\lambda_g, \mathcal{G}, \mathcal{Y})}{F(mp)}| \leq O(m^{-1/3}), \tag{5.12}$$

holds with probability $(1 - 2e^{-2n^{1/3}})$.

Claim 13. *Suppose that* $F(x) = \int_1^x P(r)dr$ *and* $y'(s) = \Pr_{G \sim P_G}[Y = 1 \mid \hat{\lambda}_G = s]$ *is Hölder continuous with constants* $a, C > 0$ *in* $s \in [0, 1]$. *Then,*

$$|\sum_{r=1}^{m} y'(1 - r/m)P(r) - \mathcal{N}(m)| \leq Cm^{-a/3}F(m) + 10. \tag{5.13}$$

Proof. Let $\mathcal{G}$ be the instance space and $g_1, ..., g_m (g_i \in \mathcal{G})$ be the m locations i.i.d. drawn from underlying distribution P_G. Let $x_{(r)} = \hat{\lambda}_{g_{(r)}}$ and we have $x_{(1)} \geq x_{(2)} \geq ... \geq x_{(m)}$ by definition. According to the Chernoff bound which is a special case of Bernstein inequalities, for each r we have $|x_{(r)} - (1 - r/m)| > m^{-1/3}$ with probability $Q = 2e^{-2m^{1/3}}$. Then, a union bound over r yields

$$\Pr[\forall r \in [m], |x_{(r)} - (1 - r/m)| \leq n^{-1/3}] \geq 1 - mQ. \tag{5.14}$$

Since $y'(s)$ is Hölder continuous with constants $a, C > 0$ in $s \in [0,1]$, we have:

$$\Pr[|\sum_{r=1}^{m}(y'(x_{(r)})P(r)-y'(1-r/m)P(r))|\leq Cm^{-a/3}\sum_{r=1}^{m}P(r)]\geq 1-mQ. \tag{5.15}$$

Considering theorem 13 and eq. (5.15) together, we obtain:

$$\Pr[|\sum_{r=1}^{m} y'(x_{(r)})P(r) - \mathcal{N}(m)| \leq 2Cm^{-a/3}F(m) + 10] \geq 1 - mQ. \tag{5.16}$$

Considering the fact that $y'(s) = \Pr_{G\sim P_G}[Y = 1 \mid \hat{\lambda}_G = s] = \mathbb{E}[Y|\hat{\lambda}_G = s]$, hence $\sum_{r=1}^{m} y'(x_{(r)})P(r)$ is the expectation of the DCG$(\lambda_g, \mathcal{G}, \mathcal{Y}) = \sum_{r=1}^{m} y_{(r)}P(r)$ conditioned on $x_{(1)}, ..., x_{(m)}$. Note that conditioning on $x_{(1)}, ..., x_{(m)}$, $y_{(r)}(r = 1, ...m)$ are independent. Thus, since that $g_1, ...g_m$ are arbitrary and for $\forall r$, $(P(r))^2 \leq P(r)$, by applying Hoeffding's inequality which is another special case of Bernstein inequalities, we have for $\forall\epsilon > 0$,

$$\Pr[|\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y}) - \sum_{r=1}^{m} y'(x_{(r)})P(r)| \geq \epsilon] \leq 2\exp(-\frac{2\epsilon^2}{F(m)}). \tag{5.17}$$

Let $\epsilon = F(m)^{2/3}$ and combine eq. (5.16) and eq. (5.17), we have

$$\Pr[|\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y}) - \mathcal{N}(m)| > 2Cm^{-a/3}F(m) + 2F(m)^{2/3}] \leq mQ + 2e^{-2F(m)^{1/3}}. \tag{5.18}$$

Thus,

$$\Pr[|\frac{\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y})}{F(mp)} - \mathcal{N}(m)| \geq 4Cp^{-1}m^{-\min(a/3,1)}] \leq mQ + 2e^{-2F(m)^{1/3}}, \tag{5.19}$$

and the theorem 10 is proved by combining theorem 12 and eq. (5.19). □

Proof of Lemma 8

We first quote two propositions from [85].

Proposition 14. *(Claim 29 at [85]) Given a fixed integer* $k \in \mathbb{N}^* = \{0\} \cup \mathbb{N}$. *For sufficiently* *large n,*

$$\int_{\frac{2}{n}}^{1} \frac{|\log^k x|dx}{(\log(nx))^{k+1}} \leq O(\frac{1}{\log^{k+1} n}), \tag{5.20}$$

and

Proposition 15. *(Claim 30 at [85]) Span($\{log^k x\}_{k\geq 0}$), is dense in $L^2[0,1]$.*

Proof. Let $\Delta y'(s) = y'(s) - \tilde{y}'(s)$. Note that $F(mp) = Li(mp+1)$, where $Li(\cdot)$ is the offset logarithmic integral function and has the property $Li(n) \sim \frac{n}{\log n}$. Hence, by the definition of the normalized pseudo expectation $\mathcal{E}(m)$ in theorem 9 and the fact that $|\Delta y'(s)| \leq 1$, we obtain:

$$\begin{aligned}\mathcal{E}(m) - \tilde{\mathcal{E}}(m) &= \frac{m}{Li(mp+1)} \int_{\frac{1}{m}}^{1} \frac{\Delta y'(1-s)ds}{\log(1+ms)} \\ &= \frac{m}{Li(mp+1)} \int_{\frac{2}{m}}^{1} \frac{\Delta y'(1-s)ds}{\log(1+ms)} + O(\frac{1}{Li(m)}).\end{aligned} \tag{5.21}$$

By expanding $\frac{1}{\log(1+ms)}$ at ms, we have:

$$|\int_{\frac{2}{m}}^{1} \frac{\Delta y'(1-s)ds}{\log(1+ms)} - \int_{\frac{2}{m}}^{1} \frac{\Delta y'(1-s)ds}{\log m + \log s}| \leq \int_{\frac{2}{m}}^{1} \frac{ds}{ms\log^2(ms)} \leq O(\frac{\log m}{m}), \tag{5.22}$$

and by expanding $\frac{1}{\log m + \log s}$ at $\log m$, we have the following

$$\begin{aligned}&|\int_{\frac{2}{m}}^{1} \frac{\Delta y'(1-s)ds}{\log m + \log s} - \sum_{z=1}^{u} \frac{(-1)^{z-1}}{\log^z m} \int_{\frac{2}{m}}^{1} \Delta y'(1-s)\log^{z-1} s ds| \\ =&|\int_{\frac{2}{m}}^{1} \frac{\Delta y'(1-s)\log^u s ds}{(\log m + \varepsilon_{m,s})^{u+1}}| \leq \int_{\frac{2}{m}}^{1} \frac{|\Delta y'(1-s)\log^u s|ds}{(\log m + \log s)^{u+1}} \leq O(\frac{1}{\log^{u+1} m})\end{aligned} \tag{5.23}$$

holds for $\forall u \in \mathbb{N}^*$, where $\varepsilon_{m,s} \in (\log s, 0)$ and we obtain the last inequality by theorem 14. Also, by theorem 15 we known that unless $\Delta y'(s) = 0$ almost everywhere, there exist a constant $k \in \mathbb{N}^*$ and a non-zero constant B so that

$$(-1)^k \int_0^1 \Delta y'(1-s)\log^k s ds = 0. \tag{5.24}$$

Assume K is the smallest k satisfying eq. (5.24) and note that

$$\int_0^{\frac{2}{n}} \log^k x dx = k! \sum_{i=0}^{k} (-1)^{k-i} \frac{x \log^i x}{i!} \Bigg|_0^{\frac{2}{n}} = O(\frac{\log^k n}{n}), \tag{5.25}$$

we finally have the following inequality by combining all the equations above:

$$|\mathcal{E}(m) - \tilde{\mathcal{E}}(m) - \frac{B}{\log^K m}| \leq O(\frac{\log^K m}{m}) + O(\frac{1}{\log^{K+1} m}).,\tag{5.26}$$

and that completes the proof of theorem 11. □

5.3.4 Proofs of the Claims

Proof of Claim 9

Proof. Let $w = \sum_{(g_i,y_i)}^{m} I[y_i = 1]$ represent the number of $y_i = 1$ in the dataset. Considering it is sampled i.i.d. and $\Pr[Y = 1] = p$, by Chernoff bound we obtain:

$$\Pr[|w/m - p| > m^{-1/3}] \leq 2e^{-2m^{1/3}}. \tag{5.27}$$

Hence, with probability larger than $1 - 2e^{-2m^{1/3}}$, we have:

$$\begin{aligned}&|\text{NDCG}(\lambda_g, \mathcal{G}, \mathcal{Y}) - \frac{\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y})}{F(mp)}| \leq |\frac{\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y})}{w} - \frac{\text{DCG}(\lambda_g, \mathcal{G}, \mathcal{Y})}{F(mp)}| \leq \\ &\text{DCG}(\lambda_g,\mathcal{G},\mathcal{Y}) \cdot \max(|\frac{1}{F(m(p-m^{-1/3}))} - \frac{1}{F(mp)}|, |\frac{1}{F(m(p+m^{-1/3}))} - \frac{1}{F(mp)}|).\end{aligned} \tag{5.28}$$

Based on the observation that $\text{DCG}(\lambda_g,\mathcal{G},\mathcal{Y}) \leq F(m)$ and the Taylor expansion of $\frac{1}{F(m(p\pm m^{-1/3}))}$ at mp, theorem 12 is proved. □

Proof of Claim 10

Proof. Based on the fact that $|P'(r)|$ and $P(r)$ are monotone decreasing functions and $P(1) + |P'(1)| < 10$, we have:

$$
\begin{aligned}
&|\sum_{r=1}^{m} y'(1-r/m)P(r) - \mathcal{N}(m)| = |\sum_{r=1}^{m} y'(1-r/m)P(r) - \int_1^m y'(1-s/m)P(s)ds| \\
=& |\sum_{r=1}^{m-1} \int_r^{r+1} \left(y'(1-r/m)P(r) - y'(1-s/m)P(s)\right) ds| + y'(0)P(m) \\
\leq& |\sum_{r=1}^{m-1} \int_r^{r+1} y'(1-s/m)(P(r) - P(s))ds| \\
&+ \sum_{r=1}^{m-1} \int_r^{r+1} |y'(1-r/m) - y'(1-s/m)|P(r)ds + y'(0)P(m) \\
\leq& \sum_{r=1}^{m-1} \int_r^{r+1} |P(r) - P(s)|ds + Cm^{-a/3} \sum_{r=1}^{m-1} P(r) + P(m) \\
\leq& \sum_{r=1}^{m-1} |P'(r)| + Cm^{-a/3}F(m) + P(m) \leq Cm^{-a/3}F(m) + |P'(1)| + \sum_{r=2}^{m} |P'(r)| + P(m) \\
\leq& Cm^{-a/3}F(m) + |P'(1)| + P(1) - P(m) + P(m) \leq Cm^{-a/3}F(m) + 10. \qquad (5.29)
\end{aligned}
$$

□

Remark: theorem 8 provides the consistent analysis of our **List-wise Fairness Criterion**. It can consistently differentiate two group in the ranking list provided by the intensity function. Thus, we consider using it to penalize the objective function later in section 5.4. By minimizing our **List-wise Fairness Criterion**, the penalties affect the final intensity function to generate a fair ranking list.

It is worth mentioning that in the Standard NDCG, the inverse logarithm function is used as the discount function. If other functions such as inverse polynomial $P(r) = r^{-\beta}, \beta > 0$ are used for computing the NDCG, the **unfairness consistency** is not exactly guaranteed. Also, an inverse polynomial decay with $\beta > 1$ might not be appropriate when the list is huge, since the tail of the ranking list may be omitted in calculation.

5.3.5 Cut-off Version

It is usually computational inhibitive the when calculate all the instance in practice. Thus, we consider a cut-off version of our **List-wise Fairness Criterion** $F(\lambda_g, \mathcal{G}, \mathcal{Y}, \tilde{\mathcal{Y}})@k$ by using the NDCG@k with $k = cm$ for some constant $c \in (0, 1)$ in eq. (5.6). We also adopt the discount function $\tilde{P}(r) = \frac{1}{\log(r+1)}$ if $r \leq k$ and $\tilde{P}(r) = 0$ otherwise. Note that it is not appropriate to define k as a constant independent with list size m. The reason is the NDCG@k is bounded by the partial summation, which cannot consistently cover the total ranking list. Thus, k must grow unboundedly when m goes to infinity. In addition, by adopting $k = cm$, the **unfairness consistency** of $F(\lambda_g, \mathcal{G}, \mathcal{Y}, \tilde{\mathcal{Y}})@k$ holds under the conditions given in theorem 8. The proof is similar to its full version in theorem 8.

5.4 Learning

In this section, we develop a penalized likelihood approach to incorporate fairness penalties into point process models. Trade-off between event prediction accuracy and fairness can be achieved by controlling the degree of fairness penalties in objective function.

5.4.1 Objective Function with List-wise Fairness Criterion

The fairness penalties based on **List-wise Fairness Criterion** for ranking grid cells with respect to sensitive groups over the total training time period $[0, T]$ is defined as follows:

$$F(\boldsymbol{\alpha}, \theta, \omega) = \frac{1}{T} \sum_{t=1}^{T} (NDCG(\lambda_g(t), \mathcal{G}, \mathcal{Y}) - NDCG(\lambda_g(t), \mathcal{G}, \tilde{\mathcal{Y}}))^2. \tag{5.30}$$

When $F = 0$, the ranking list with respect to the two groups achieves consistently fairness averagely over a time period.

More generally, suppose there are q types of sensitive features and the i-th type of sensitive features $\boldsymbol{f}_i$ contains c_i groups, then for $\forall l_i, l_i' \in c_i$, the total penalty is defined as follows:

$$F(\boldsymbol{\alpha}, \theta, \omega) = \frac{1}{T} \sum_{i=1}^{q} \sum_{l_i > l_i'} \sum_{t=1}^{T} (NDCG(\lambda_g(t), \mathcal{G}, \mathcal{Y}_{l_i}) - NDCG(\lambda_g(t), \mathcal{G}, \mathcal{Y}_{l_i'}))^2, \tag{5.31}$$

where $\mathcal{Y}_{l_i}$ is the l-th group of the i-th type sensitive features. For example, sensitive features include race and gender. There are multiple types of race and different gender. When $F = 0$, for every type of sensitive features and for each pair of feature groups, the ranker achieves consistently fairness averagely over a time period.

Finally, we add the penalty F weighted by a trade-off parameter γ to the objective function eq. (5.3) and minimize:

$$OPT = \min \mathcal{L}(\boldsymbol{\alpha}, \theta, \omega) + \rho\{T^{-1} \sum_{t=1}^{T} tr(\boldsymbol{\Lambda}(t)^{\top} L \boldsymbol{\Lambda}(t))\} + \gamma F(\boldsymbol{\alpha}, \theta, \omega). \tag{5.32}$$

Once we obtain $\boldsymbol{\alpha}, \theta$ and ω, we can directly calculate the intensities for all grid cells by eq. (5.1) and present a fair ranking list.

5.4.2 Optimization and Scalability

The objective function with fairness penalties defined by eq. (5.32) is non-differentiable since grid cells needs to be ranked by intensities and a threshold is required for selecting top-k cells at each time slot t. Thus, we adopt the Nelder-Mead simplex method [48] to find a local minimum and the method works well in practice. We show the details how we apply this method in section 5.5.2.

It is well known that simplex method takes polynomial time complexity, i.e., $O(n^k)$ in average [77], which is computational inhibitive when the dataset is huge. Hence, we provide a smoothed variation of our method, which uses a non-linear function to approximate the rank and makes it differentiable.

As we introduced before, for the standard NDCG, we use the inverse logarithm decay $P(r) = \frac{1}{\log(r+1)}$ as the discount function. We first rewrite the standard DCG in the following form:

$$DCG(\lambda_g, \mathcal{G}, \mathcal{Y}) = \sum_{i=1}^{m} \frac{y_i}{\log(R(i)+1)}, \tag{5.33}$$

where $R(i)$ is the rank position of the cell g_i by the ranker, intensity function λ_g. The DCG is non-smooth mainly because of the non-continuous mapping from the intensity score λ_{g_i} to the rank position $R(i)$. Specifically, the rank position can be defined in the following form:

$$R(i) = 1 + \sum_{j \neq i} I_{\{\lambda_{g_i} - \lambda_{g_j} < 0\}}. \tag{5.34}$$

To deal with this problem, we follow [68] to revise the discount function so that it becomes a continuous function of the intensities. Thus, we have the approximate rank position $\tilde{R}(i)$ as:

$$\tilde{R}(i) = 1 + \sum_{j \neq i} \frac{\exp(-\delta(\lambda_{g_i} - \lambda_{g_j}))}{1 + \exp(-\delta(\lambda_{g_i} - \lambda_{g_j}))}, \tag{5.35}$$

where δ is the hyper-parameter which is often set dynamically like the decay of learning rate. A larger δ leads to a better approximation of rank position. However it increase the difficulty of optimization due to the stronger degree of nonlinearity. When $\lambda_{g_j} \ll \lambda_{g_i}$, the non-linear part approaches zero, thus the position hardly changes. Integrating the approximate rank position eq. (5.35) into eq. (5.33), we obtain the smoothed DCG. The smoothed DCG can be optimized by gradient based methods, which makes the computation faster and the model scalable. Nevertheless, we have to mention that this smoothed method is *not* suitable for the cut-off version NDCG@k. In addition, the smoothed method *cannot* guarantee the **unfairness consistency** property we introduced before.

It is worth mentioning that our **List-wise Fairness Criterion** is not limited to spatial-temporal point processes, in fact, it can be extended to other ranking problems. For example, suppose we recommend a candidate list and the candidates may have sensitive features such as gender and race. Our fairness metric can be applied to either binary or finite sets of features. The computational complexity increases when the list is huge (e.g., a million). In this case, our method using the smoothed DCG can tackle the computation challenge and we can obtain an approximately fair ranking list.

5.5 Experiment

In this section, we introduce the experiments and results.

5.5.1 Data

We evaluate our list-wise fairness criterion on three open sourced real-world datasets detailed in table 5.2. Specifically, the **Portland** dataset [4] [61] is provided by 2017 NIJ Crime Forecasting Challenge [5] that tasks participants to predict the spatial locations with highest numbers of crime related calls in Portland, OR. It contains a list of events with geographic coordinates, timestamps, and the types of events such as burglary, street crime, and auto theft from March, 1, 2012 to February 28, 2017. In our setting, a unit time slot t is a day. Each event is assigned to one of equal sized regular rectangle grids based on the longitude and latitude. In the experiments, we only use the street event data and we simulate the race populations for white and Hispanic/Latino as the sensitive features, which is an extreme case. We first learn the model without fairness penalties to obtain a ranked list of locations, and assign the population for white as 1 to m in the order from high to low and m to 1 for the Hispanic/Latino. The **Dallas** dataset [6] in Kaggle comes from the Dallas Police Department containing detailed incidence reports for around 3 years at Dallas, Texas. We adopt the similar settings for Portland dataset to specify the locations that the events belong to. For the race population feature, we count the number of events for complainants in three races (black, white, and Hispanic/Latino) and regard them as the population of that location grid. The **Houston** dataset [7] is a crowdsourcing dataset obtained from a Google doc which contains rescue requests for 3 days around Harris County in Greater Houston Area during the Hurricane Harvey disaster. In this dataset, a time slot t is an hour and we use the ZIP Code as the location id. We get the race population statistics from American FactFinder [8]. We use the populations of white and Hispanic/Latino as the sensitive features.

For Portland and Dallas datasets, we use the first 200 days for training and the days from 201 to 400 for testing given the huge number of events, while for Houston dataset we adopt the first 14 hours for training and the rest for testing. For geometric settings such as graph regularizers

[4]https://github.com/gomohler/crimerank
[5]https://nij.ojp.gov/funding/real-time-crime-forecasting-challenge
[6]https://www.kaggle.com/carrie1/dallaspolicereportedincidents
[7]https://data.world/sya/harvey-rescue-doc
[8]https://factfinder.census.gov/faces/nav/jsf/pages/index.xhtml

Table 5.2: Dataset description.

Dataset	Events	Geo-Type	Unique-IDs	Time	Groups
Portland	$166K$	Grid	398	$1916d$	2
Dallas	$201K$	Grid	303	$853d$	3
Houston	1182	ZIP Code	106	$26h$	2

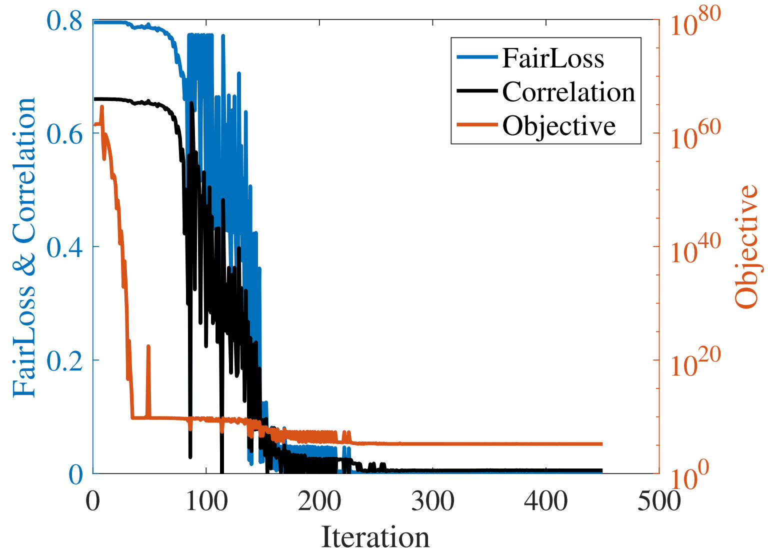


Figure 5.1: Convergence of the algorithm on Portland dataset.

in eq. (5.32), we assume that each location is a node in a graph and adjacent location nodes are connected. In training, we use cut-off version of our list-wise fairness criterion as the fairness penalties with NDCG@50 to improve computational efficiency. Since the optimization algorithm only converges to local minimal, we run several times with different initialization and present the best results.

5.5.2 Experiment Setting and Algorithm Convergence

We apply the Nelder-Mead simplex method in $MATLAB^{®}$ by using the function ”fminsearch” [9], which can find local minimum of unconstrained multivariable functions using derivative-free method. Specifically, the code contains three parts: a script file that set all the variables and initials such that apply the ”fminsearch” over objective function; a function file that is exactly the objective; and another function file calculates the log-likelihood, graph regularizer and the fairness penalties. We initialize the parameter as $\boldsymbol{\alpha} = \mathbf{0}$, $\theta = 0.8$, and $\omega = e^{-2}$ for all the

[9]https://www.mathworks.com/help/matlab/ref/fminsearch.html

datasets. We set geometric trade-off parameter $\rho = 1$ and vary the fairness trade-off parameters as $\gamma = 10^s$, $s = [0, 1, 2, ..., 8]$. We follow [62] to define the *fair* model learned with $\gamma = 10^8$ and the neutral model without fairness penalties($\gamma = 0$). The other experiments settings about datasets are introduced in section 5.5.1.

We show the convergence curve of the algorithm in fig. 5.1 on Portland Dataset. We can see that the method works well and the fairness loss, correlation and the value of objective finally stably converge to a local minimal.

5.5.3 Evaluation Metrics

We use several metrics to evaluate both prediction performance and fairness influence by adopting our list-wise fairness criterion.

Fairness Evaluation Metrics

• **NDCG@k**: We directly compare the NDCG@50 scores of different groups since we use them in training. A smaller difference indicates a fairer prediction.

• **FairLoss**: We apply the fairness penalties that we define in eq. (5.32) to the test data. A smaller fairness loss indicates a fairer ranking list.

• **Patrol@k**: We use the fairness metric defined in [62], which is the ratio of the summation of population in the top-k locations over that in the total list per race group. Specifically,

$$\text{Patrol@k} = \frac{\sum_{i=1}^{k} y_{(i)}}{\sum_{i=1}^{m} y_m}, \tag{5.36}$$

and we just replace the NDCG with this metric and the difference between two groups should still be averaged over time slot t and types i in eq. (5.31). We name it **List-sum Fairness Criterion**, in contrast to our **List-wise Fairness Criterion**. In the experiments, we adopt $k = 50$ to keep uniform standard with the former setting.

Prediction Performance Evaluation Metrics

• **Correlation**: We use the Pearson correlation coefficient between the predicted intensity list $\boldsymbol{\Lambda}(t)$ and the ground truth, which is the list of numbers of events at time slot t, to evaluate the prediction performance. It is between 1 and -1, where 1 indicates total positive linear correlation,

0 means no linear correlation, and -1 represents total negative linear correlation according to the Cauchy–Schwarz inequality.

• **TestLoss**: It is the test loss without fairness penalties in eq. (5.32), which is the log-likelihood of point process in eq. (5.2) that indicates the probability that existing history event $\mathcal{T}$ has happened and no events happen in $[t_n, t)$.

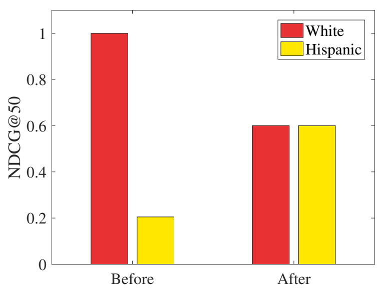
• **PAI@k**: Predictive Accuracy Index (PAI) is widely used to measure the percentage of crime events in the top-k locations [61, 62, 12, 64] and has the following form:

$$\text{PAI@k} = \frac{\text{events in k locations}}{\text{total events}} \cdot \frac{\text{total area}}{\text{area of k locations}}. \tag{5.37}$$

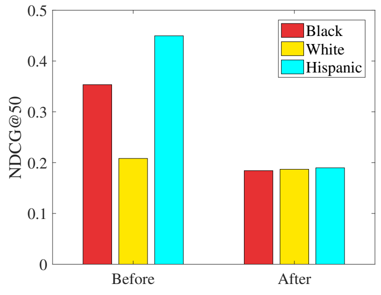
Since PAI@k is area normalized, a value of 1 indicates random predictions. We also apply it to the Houston rescue dataset. The value of k is chosen by the police resources or the rescue resources and we provide two choices in the experiments, PAI@15 and PAI@50.

5.5.4 Fairness over Groups

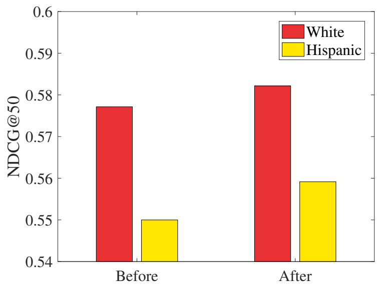
We plot both the neutral (before adding our list-wise fairness penalties) and fair scenarios of our model by measuring NDCG@50 on test data per group over all three datasets in fig. 5.2. We can see that in general, this list-wise fairness criterion is effective and the differences between the groups become smaller after adding our list-wise fairness penalties, and all the scores become closer to each other to approach the ideal case with the fairness penalties close to 0. The performance on Houston dataset is not as good as the former two due to data sparsity. In particular, there are much fewer events in a little smaller number of unique ZIP Codes as described in table 5.2. Besides the time slot t is an hour, and thus the events/time that represents the temporal sparsity is also at a low level. Therefore, the locations most influenced by Hurricane Harvey might have much higher intensities and much more rescue requests than others. As a result, it requires a much larger fairness penalty to change the order of the ranking list. This leads to the weak performance in terms of the fairness metrics and makes it hard to balance the NDCG@50 values between two groups. In addition, for the neutral scenario, the difference of the NDCG@50



(a) Portland



(b) Dallas



(c) Houston

Figure 5.2: The NDCG@50 for different racial groups before and after adding list-wise fairness penalties.

values between two groups for Houston data is relatively smaller than others, which indicates the fairness penalty unscaled with γ is relatively small. That also increases the difficulty in obtaining an extremely fair ranking list.

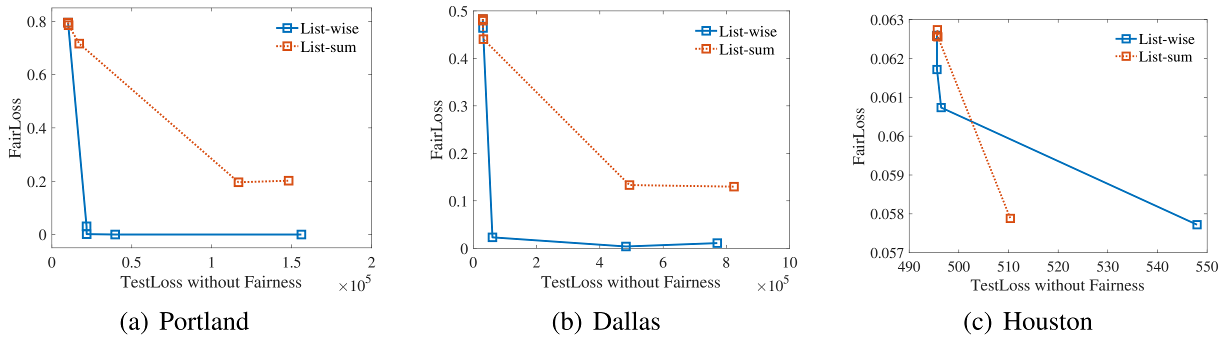
5.5.5 Fairness vs. Prediction Performance

We measure the prediction performance and present the correlation and PAI@k before and after adding our list-wise fairness penalties in table 5.3. A higher correlation coefficient indicates stronger correlation between the predicted intensity list and the ground truth of the number of events, which finally represents the point process model's prediction accuracy. A higher PAI@k does not reflect the ranking accuracy of the intensity list according to the definition; however, it represents a higher predicted number of independent events at top-k locations which is useful in practice with limited police and rescue resources.

From the table, we can see that at first, the prediction performance is influenced when we incorporate the fairness in objective functions. The ranking prediction performance represented by the correlation is affected to a large extent. However, either PAI@15 or PAI@50 still keeps a higher level. This demonstrates that most of the hotspots is still on the top of the predicted ranking list. It is worth mentioning that although there is a significant cost in considering the list-wise fairness, the PAI value is not only much higher than the random case, which is 1, but also potentially even more accurate than human analysis.

Table 5.3: Average prediction performance before and after adding list-wise fairness penalties.

Dataset	Accuracy Measure	Results	
		Before	After
Portland	PAI@15	**344.0795**	263.9681
	PAI@50	**194.2702**	95.9875
	Correlation	**0.6614**	0.0030
Dallas	PAI@15	**156.3209**	21.1041
	PAI@50	**105.2752**	16.4861
	Correlation	**0.6550**	0.1534
Houston	PAI@15	**455.4241**	400.3325
	PAI@50	**179.1580**	171.0044
	Correlation	**0.3993**	0.3367



(a) Portland (b) Dallas (c) Houston

Figure 5.3: Fairness-accuracy curves for list-wise and list-sum fairness.

5.5.6 Comparison between List-wise and List-sum Fairness

Similar to [5], we investigate the trade-off between accuracy and fairness for two different types of fairness penalties including the **List-wise Fairness** and **List-sum Fairness**. We apply these two different fairness metrics in the training stage, and adjust the trade-off parameter of the fairness γ in the range 10^s, $s = [0, 1, 2, ..., 8]$. The x axis is the test loss without considering fairness penalties and it indicates the model prediction performance. A lower test loss value represents better prediction performance. The y axis is the fairness penalty based on our list-wise fairness criterion and is calculated over test data. A lower value means a fairer ranking list.

According to the results shown in fig. 5.3, we can see that for all the three datasets, the degree of the fairness of the model increases as the trade-off parameter γ becomes larger, resulting in a worse prediction. Also, with the same level of the fairness loss, our **List-wise Fairness** achieves

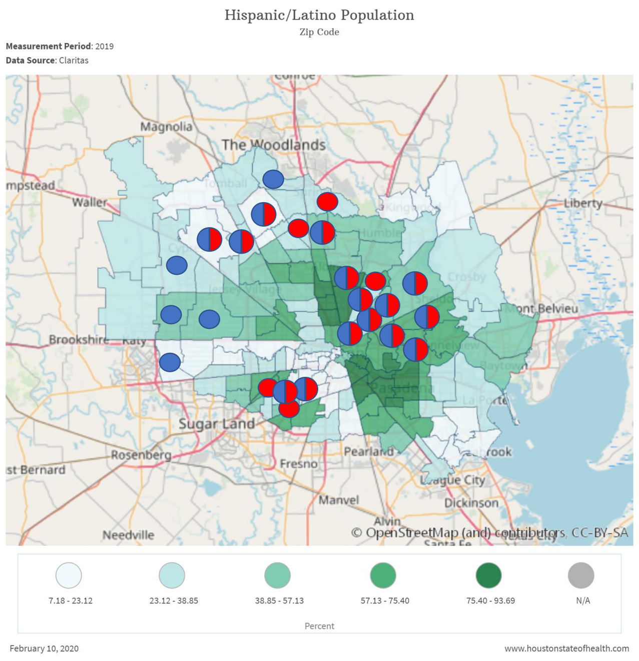


Figure 5.4: Case study for Hispanic/Latino population.

better prediction performance than the **List-sum Fairness** in general. This indicates that our list-wise fairness criterion is more efficient and less costly in the optimization. In addition, note that the fairness loss for Houston data is relatively smaller than others as we described in section 5.5.4. List-wise fairness have resulted in consistently more efficient curves with different values of trade-off parameter γ than list-sum fairness.

5.5.7 Case study

We visualize the top-20 detected hotspots before (blue) and after (red) adding our list-wise fairness penalties in fig. 5.4. The circle of both blue and red indicates that the location are captured in both ranking lists. The background [10] shows the population of Hispanic/Latino ranked by percentage at Harris County in the Greater Houston Area, Texas. A total of 5 locations has changed in the top-20 list and it is obvious that they switch to the locations with more Hispanic/Latino population in general. Even though the Houston dataset is sparse and it is hard to obtain a fair ranking list as we introduced in section 5.5.4, the results are still visible in the figure. It is worth mentioning that the east of Harris County, where several hotspots are detected in both

[10]Downloaded from the website: http://www.houstonstateofhealth.com/

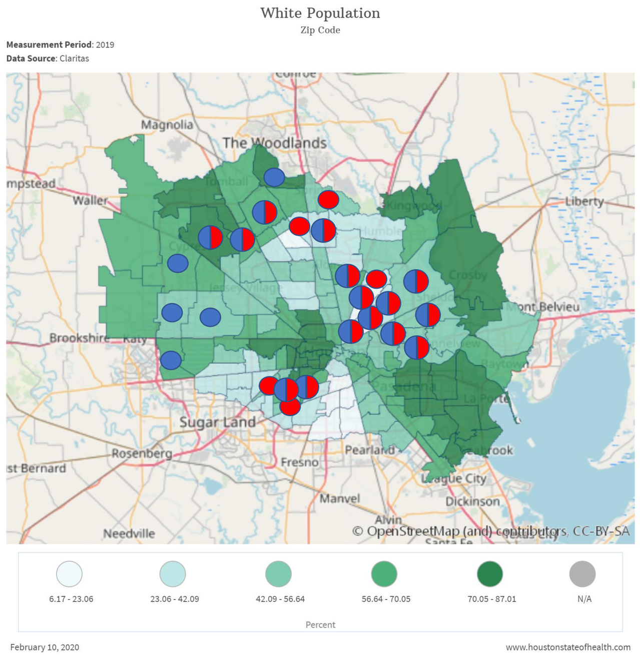


Figure 5.5: Case study for White population.

neutral and fair ranking lists, is the worst-hit area suffering Hurricane Harvey. Since the model still keeps these top predicted locations, it demonstrates the effectiveness of the spatial-temporal point process in predicting the future events. Similar results are obtained on the white population map and presented in fig. 5.5.

CHAPTER 6
CONCLUSIONS

6.1 Summary

In this research proposal, we present several novel point process-based models and learning algorithms to analyze the asynchronous event data in various domains such as social networks, business directory services, TV systems, and the job markets. Specifically, we've made the following contributions:

• **Mutual Influences Modeling via Local Models.** In Chapter 2, we present a novel framework that integrates the kernel smoothing and the Hawkes process to model the temporal events of user-item interactions. We assume that the intensity parameter matrix is locally low-rank. With non-parametric kernel smoothing, each user-item pair can be simulated by a series of local matrix mappings. We design an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art. This model can be applied to other 2D aggregated Hawkes processes, such as temporal user interactions in social networks, and extended to n-dimensional aggregated Hawkes processes, as long as these dimensions satisfy the local low-rank assumption.

• **Integrating Geometric Structure with Point Process.** In Chapter 3, we present a novel framework that integrates the graph convolutional recurrent neural network and Hawkes processes to model temporal events. This 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.

• **Fairness-aware Point Process Models with Graph Embedding.** In Chapter 4, we present an efficient point process framework that incorporates geometric structure with coevolving nature of feature embedding to tackle data sparsity, and introduce several novel fairness metrics that penalize the event likelihood function to enforce fairness. Extensive experiments on the real world datasets demonstrate that our method can not only benefit event prediction but also balance between accuracy and fairness.

• **List-wise Fairness Criterion for Point Processes.** In Chapter 5, we present a novel list-wise fairness criterion to obtain a fair ranking list for predicting top-k locations via spatial-temporal point process. We propose a strict definition of the unfairness consistency property of a fairness metric and prove that our list-wise fairness criterion satisfies this property. Extensive experiments on the real-world datasets demonstrate the effectiveness of the list-wise fairness criterion.

In summary, our models and learning algorithms effectively extract the hidden dynamics between the complicated asynchronous event data and successfully tackle not only the sparsity but also the amplified self-excitation bias problems. The experiment results on real-world dataset demonstrate their practicability in diverse applications.

6.2 Discussions and Future Research Directions

First of all, we currently only incorporate point process-based models with shallow neural network contains several layers. We'd like to integrate point processes with different types of deeper neural network structures such as transformers and residual neural networks. Secondly, we believe that point process-based models can be extended to other application areas especially natural language processing. Thirdly, for the geometric structure embedding, it is obvious that graph structure in the real world is dynamic, thus developing novel techniques with dynamic graph embedding is a big opportunity. Last but not least, we'd like to extend our research on fairness-aware point processes, for example, adopting individual fairness or preprocessing sensitive features before optimization, or developing other novel scalable fairness metrics for large scale datasets and more efficient optimization methods of pairwise and top-K fairness metrics.

REFERENCES

[1] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. *Survival and Event History Analysis: A Process Point of View*. Springer Science & Business Media, 2008.

[2] Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005, 2015.

[3] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California Law Review*, 104:671, 2016.

[4] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 95–104, 2007.

[5] Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. A convex framework for fair regression. *arXiv preprint arXiv:1706.02409*, 2017.

[6] Wim Bernasco, Shane D Johnson, and Stijn Ruiter. Learning where to offend: Effects of past on future burglary locations. *Applied Geography*, 60:120–129, 2015.

[7] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. Fairness in recommendation ranking through pairwise comparisons. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2212–2220, 2019.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[9] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *Proc. of the IEEE International Conference on Data Mining Workshops*, pages 13–18, 2009.

[10] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. Ranking with fairness constraints. *arXiv preprint arXiv:1704.06840*, 2017.

[11] Guido Cervone, Elena Sava, Qunying Huang, Emily Schnebele, Jeff Harrison, and Nigel Waters. Using twitter for tasking remote-sensing data collection and damage assessment: 2013 Boulder flood case study. *International Journal of Remote Sensing*, 37(1):100–124, 2016.

[12] Spencer Chainey, Lisa Tompson, and Sebastian Uhlig. The utility of hotspot mapping for predicting spatial patterns of crime. *Security journal*, 21(1-2):4–28, 2008.

[13] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[14] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[16] Stéphan Clémençon, Gábor Lugosi, Nicolas Vayatis, et al. Ranking and empirical minimization of u-statistics. *The Annals of Statistics*, 36(2):844–874, 2008.

[17] David R Cox and Peter Adrian Walter Lewis. Multivariate point processes. In *Proc. 6th Berkeley Symp. Math. Statist. Prob*, volume 3, pages 401–448, 1972.

[18] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016.

[19] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le 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)*, pages 1555–1564, 2016.

[20] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. Time-sensitive recommendation from recurrent user activities. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3492–3500, 2015.

[21] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proc. of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226, 2012.

[22] Michael Eichler, Rainer Dahlhaus, and Johannes Dueck. Graphical modeling for multivariate Hawkes processes with nonparametric link functions. *Journal of Time Series Analysis*, 38(2):225–242, 2017.

[23] Jalal Etesami, Negar Kiyavash, Kun Zhang, and Kushagra Singhal. Learning network of multivariate Hawkes processes: a time series approach. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 162–171, 2016.

[24] Mehrdad Farajtabar, Nan Du, Manuel Gomez Rodriguez, Isabel Valera, Hongyuan Zha, and Le Song. Shaping social activity by incentivizing users. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2474–2482, 2014.

[25] Mehrdad Farajtabar, Yichen Wang, Manuel Gomez Rodriguez, Shuang Li, Hongyuan Zha, and Le Song. Coevolve: A joint point process model for information diffusion and network co-evolution. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1954–1962, 2015.

[26] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 259–268, 2015.

[27] Eric W Fox, Martin B Short, Frederic P Schoenberg, Kathryn D Coronges, and Andrea L Bertozzi. Modeling e-mail networks and inferring leadership using self-exciting point processes. *Journal of the American Statistical Association*, 111(514):564–584, 2016.

[28] Andrew M Freed. Earthquake triggering by static, dynamic, and postseismic stress transfer. *Annu. Rev. Earth Planet. Sci.*, 33:335–367, 2005.

[29] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research (JMLR)*, 4(Nov):933–969, 2003.

[30] Debarghya Ghoshdastidar and Ambedkar Dukkipati. On power-law kernels, corresponding reproducing kernel hilbert space and applications. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2013.

[31] Prem Gopalan, Jake M Hofman, and David M Blei. Scalable recommendation with hierarchical poisson factorization. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 326–335, 2015.

[32] Eric C Hall and Rebecca M Willett. Tracking dynamic point processes on networks. *IEEE Transactions on Information Theory*, 62(7):4327–4346, 2016.

[33] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[34] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3315–3323, 2016.

[35] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[36] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[37] Balázs Hidasi and Domonkos Tikk. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, 30(2):342–371, 2016.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Zubin Jelveh and Michael Luca. Towards diagnosing accuracy loss in discrimination-aware classification: An application to predictive policing. *Fairness, Accountability and Transparency in Machine Learning*, 26(1):137–141, 2014.

[40] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[41] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

[42] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *Proc. of the IEEE 11th International Conference on Data Mining Workshops*, pages 643–650, 2011.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[44] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[45] Michael Krumin, Inna Reutsky, and Shy Shoham. Correlation-based analysis and generation of multiple spike trains using Hawkes models with an exogenous input. *Frontiers in Computational Neuroscience*, 4:147, 2010.

[46] Juhi Kulshrestha, Motahhare Eslami, Johnnatan Messias, Muhammad Bilal Zafar, Saptarshi Ghosh, Krishna P Gummadi, and Karrie Karahalios. Quantifying search bias: Investigating sources of bias for political searches in social media. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 417–432, 2017.

[47] Takashi Kurokawa, Taihei Oki, and Hiromichi Nagao. Multi-dimensional graph fourier transform. *arXiv preprint arXiv:1712.07811*, 2017.

[48] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

[49] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

[50] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

[51] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 82–90, 2013.

[52] Joonseok Lee, Mingxuan Sun, Seungyeon Kim, and Guy Lebanon. Automatic feature induction for stagewise collaborative filtering. In *Proc. of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 314–322, Dec. 2012.

[53] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. PREA: Personalized recommendation algorithms toolkit. *Journal of Machine Learning Research (JMLR)*, 13(1):2699–2703, Sep. 2012.

[54] Rémi Lemonnier, Kevin Scaman, and Argyris Kalogeratos. Multivariate Hawkes processes for large-scale inference. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 2168–2174, 2017.

[55] Scott Linderman and Ryan Adams. Discovering latent network structure in point process data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1413–1421, 2014.

[56] Thomas Josef Liniger. *Multivariate Hawkes Processes*. PhD thesis, ETH Zurich, 2009.

[57] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.

[58] Stephane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[59] Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 6754–6764, 2017.

[60] George Mohler, Jeremy Carter, and Rajeev Raje. Improving social harm indices with a modulated hawkes process. *International Journal of Forecasting*, 34(3):431–439, 2018.

[61] George Mohler, Michael D Porter, Jeremy Carter, and Gary LaFree. Learning to rank spatio-temporal event hotspots. In *Proceedings of the 7th international workshop on urban computing*, 2018.

[62] George Mohler, Rajeev Raje, Jeremy Carter, Matthew Valasik, and Jeffrey Brantingham. A penalized likelihood method for balancing accuracy and fairness in predictive policing. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2454–2459, 2018.

[63] George O Mohler, Martin B Short, P Jeffrey Brantingham, Frederic Paik Schoenberg, and George E Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108, 2011.

[64] George O Mohler, Martin B Short, Sean Malinowski, Mark Johnson, George E Tita, Andrea L Bertozzi, and P Jeffrey Brantingham. Randomized controlled field trials of predictive policing. *Journal of the American statistical association*, 110(512):1399–1411, 2015.

[65] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3697–3707, 2017.

[66] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[67] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[68] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13(4):375–397, 2010.

[69] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2107–2115, 2015.

[70] Manuel Gomez Rodriguez. *Structure and Dynamics of Diffusion Networks*. PhD thesis, 2013.

[71] Chris Russell, Matt J Kusner, Joshua Loftus, and Ricardo Silva. When worlds collide: Integrating different counterfactual assumptions in fairness. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 6414–6423, 2017.

[72] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[73] S Sastry. Some NP-complete problems in linear algebra. *Honors Projects*, 1990.

[74] Jin Shang and Mingxuan Sun. Local low-rank Hawkes processes for temporal user-item interactions. In *Proc. of the IEEE International Conference on Data Mining (ICDM)*, pages 427–436, Nov. 2018.

[75] Jin Shang and Mingxuan Sun. Geometric Hawkes processes with graph convolutional recurrent neural networks. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1–8, 2019.

[76] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[77] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[78] Mingxuan Sun, Guy Lebanon, and Paul Kidwell. Estimating probabilities in recommendation systems. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 734–742, Apr. 2011.

[79] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 3462–3471. JMLR. org, 2017.

[80] Rasmus Waagepetersen. Estimating functions for inhomogeneous spatial point processes with incomplete covariate data. *Biometrika*, 95(2):351–363, 2008.

[81] Brian Wallace. Constrained optimization: Kuhn-tucker conditions. 2004.

[82] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

[83] Xin Wang, Roger Donaldson, Christopher Nell, Peter Gorniak, Martin Ester, and Jiajun Bu. Recommending groups to users using user-group engagement and time-dependent matrix factorization. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2016.

[84] Yichen Wang, Nan Du, Rakshit Trivedi, and Le Song. Coevolutionary latent feature processes for continuous-time user-item interactions. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 4547–4555, 2016.

[85] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of ndcg type ranking measures. In *Proc. of the Conference on Learning Theory (COLT)*, pages 25–54, 2013.

[86] Wenming Xiao, Xiao Xu, Kang Liang, Junkang Mao, and Jun Wang. Job recommendation with Hawkes process: an effective solution for RecSys Challenge 2016. In *Proc. of the Recommender Systems Challenge*, 2016.

[87] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. Learning granger causality for Hawkes processes. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1717–1726, 2016.

[88] Hongteng Xu, Dixin Luo, and Hongyuan Zha. Learning Hawkes processes from short doubly-censored event sequences. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 3831–3840, 2017.

[89] Hongteng Xu, Weichang Wu, Shamim Nemati, and Hongyuan Zha. Patient flow prediction via discriminative learning of mutually-correcting processes. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):157–171, 2017.

[90] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–6, 2017.

[91] Yingxiang Yang, Jalal Etesami, Niao He, and Negar Kiyavash. Nonparametric hawkes processes: Online estimation and generalization bounds. *arXiv preprint arXiv:1801.08273*, 2018.

[92] Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2921–2930, 2017.

[93] Adams Wei Yu, Wanli Ma, Yaoliang Yu, Jaime Carbonell, and Suvrit Sra. Efficient structured matrix rank minimization. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1350–1358, 2014.

[94] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proc. of the International World Wide Web Conference (WWW)*, pages 1171–1180, 2017.

[95] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *Proc. of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1569–1578, 2017.

[96] Richard Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning fair representations. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 325–333, 2013.

[97] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 641–649, 2013.

[98] Lei Zou, NSN Lam, Shayan Shams, Heng Cai, Michelle A Meyer, Seungwon Yang, Kisung Lee, Seung-Jong Park, and Margaret A Reams. Social and geographical disparities in Twitter use during Hurricane Harvey. *International Journal of Digital Earth*, pages 1–19, 2018.

VITA

Jin Shang received the B.S. degree in theoretical and applied mechanics and the M.S. degree in solid mechanics, both from University of Science and Technology of China, Hefei, China, in 2013 and 2016, respectively. He is currently pursuing his Ph.D. degree in the Division of Computer Science and Engineering at Louisiana State University, Baton Rouge, LA, USA. His research interests include machine learning as well as deep learning models and algorithms, with applications in recommender systems and time-series analysis.