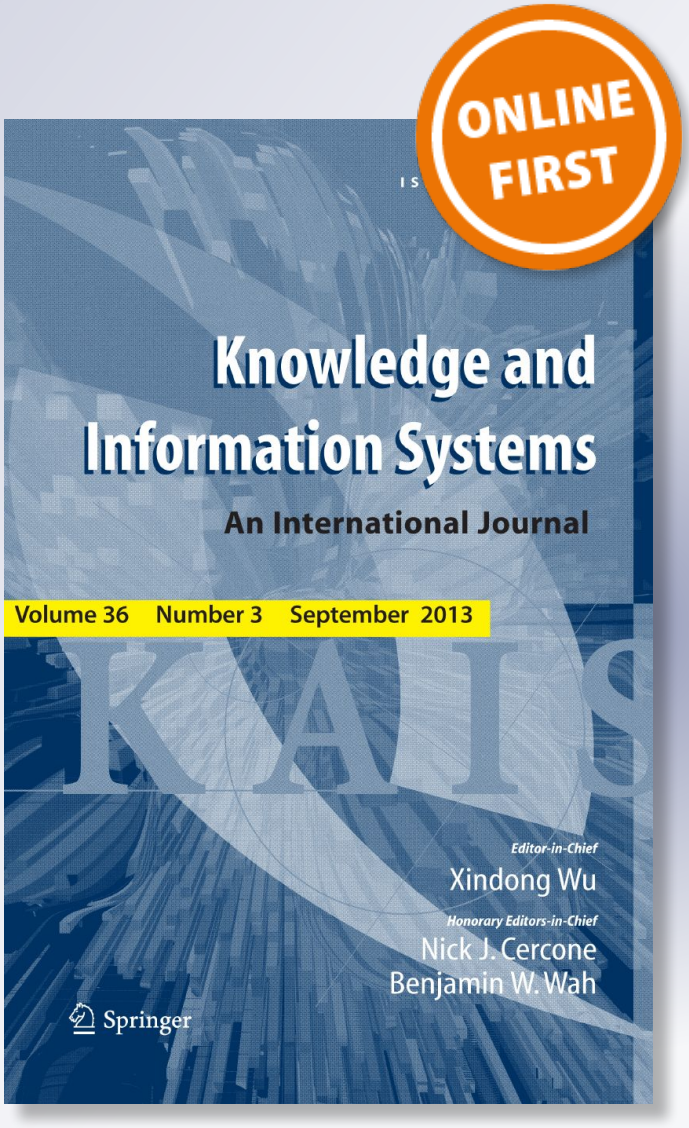
Local low-rank Hawkes processes for modeling temporal user–item interactions

Jin Shang & Mingxuan Sun

Knowledge and Information Systems
An International Journal

ISSN 0219-1377

Knowl Inf Syst
DOI 10.1007/s10115-019-01379-6



Springer

Knowledge and Information Systems
https://doi.org/10.1007/s10115-019-01379-6

REGULAR PAPER

Local low-rank Hawkes processes for modeling temporal user–item interactions

Jin Shang[1] · **Mingxuan Sun**[1]

Received: 4 January 2019 / Revised: 21 June 2019 / Accepted: 23 June 2019

Abstract
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, 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.

Keywords Hawkes process · Kernel smoothing · Sequential data

1 Introduction

Hawkes processes have become very popular in modeling recurrent user–item interaction events that exhibit mutual-excitation properties in various domains [8,29]. For example, Hawkes processes can be used to model user behaviors in online services, where the interac-

This paper is an extended version of the full paper published in the proceedings of the IEEE International Conference on Data Mining (ICDM) 2018 [21].

✉ Mingxuan Sun
msun@csc.lsu.edu

Jin Shang
jshang2@lsu.edu

[1] Division of Computer Science and Engineering, Louisiana State University, Baton Rouge, LA 70803, USA

Published online: 09 July 2019

tion 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 [5,25] 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 [5]. 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 [24] 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) [10,18] 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 ith dimension is characterized by the base intensity and the linear combination of the influences of events occurred in every other dimension on the ith dimension. Extensive research [6,7,17,26] 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 [6,9,11].

In this paper, 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 nonparametric 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.

The remainder of this paper is organized as follows: in Sect. 2, we review the related work. In Sect. 3, we introduce the local low-Rank Hawkes process. In Sect. 4, we present an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. In Sect. 5, we highlight the characteristics of the model performance in terms of the dependency on feature integration

methods, the kernel bandwidth parameter, the number of anchor points, and anchor point selection approaches. In Sect. 6, we report the results of extensive experiments on real-world datasets and demonstrate the performance improvements of our model in comparison with the state of the art.

2 Related work

Local low-rank matrix completion with kernel smoothing [14] 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 nonparametric regression of those mappings. Previous work [14] mainly focuses on two-dimensional matrix factorization without the temporal dimension. Our work is modeling a sequence of events and the objective function is completely different from the mean squared loss of ratings in matrix completion tasks. In addition, we use the trace norm to enforce the low-rank assumption and the previous work [14] explicitly decomposes a matrix to the product of two low-rank matrices. Variations of matrix completion methods [15,16,22] are widely applied in recommender systems.

Hawkes processes [10] can be used in a variety of applications such as inferring granger causality [26], modeling patient records in smart health [27], and predicting online social activities [29]. For example, a multi-dimensional Hawkes process has been proposed by Zhou et al. [29] to learn the social event diffusion in sparse low-rank networks. A multivariate Hawkes process has further been proposed by Farajtabar et al. [8] 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 [6,9,11].

For modeling large-scale user–item interactions, previous work [5] simulates the temporal events of (u, i) pair as a one-dimensional Hawkes process and assumes that all user–item pairs are independent. The intensity of each Hawkes process is estimated based solely on the individual pair's observed sequence. The performance of the method degrades when there are no sufficient observed events for an individual (u, i) pair. We consider the approach a *point to point* intensity estimation.

A coevolutionary latent feature process has been proposed in [24], which constructs inter-dependent Hawkes processes by integrating additional features such as item features, user features, and interaction features between users and items. The intensity of the events of each user–item pair is estimated by aggregating the influences of all previous events of other user–item pairs weighted by user and item similarities. Hence, the time prediction accuracy improves since a large number of events are used to simulate only one pair's intensity and a huge amount of auxiliary feature information is incorporated. However, the item rank prediction becomes worse because the individual preferences are influenced by the general preference. We consider the approach a *matrix to point* intensity estimation.

Our model is different from others in that we assume a local low-rank structure in user–item dimension, which models the intensity for one (u, i) pair as the aggregation of several neighbors. The smoothing kernels are used to evaluate the similarities between these neighbors and the target (u, i) pair. The final estimation of the target (u, i) pair's intensity integrates the influences of neighbors with different weights, which we consider a *neighbors to point* intensity estimation.

Table 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 $\mathcal{O}$
$t_i^{u,i}$	ith 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	τth entry of a list of q anchor points
s_τ	User–item anchor pair (a_τ, b_τ) for the τth anchor points
$\boldsymbol{H}^{s_\tau}$	Base intensity matrix for the τth anchor point pair s_τ
$\boldsymbol{A}^{s_\tau}$	Self-exciting coefficient matrix for the τth 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 τth 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. (8)
$\boldsymbol{M}\{u, i\}$	Vector defined in Eq. (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
$\boldsymbol{X}^{s_\tau}$	τth local model parameter block matrix $[\boldsymbol{H}^{s_\tau}; \boldsymbol{A}^{s_\tau}]$
λ^τ, β^τ	τth local model trade-off factors for constraints
ρ^τ	τth local model regularization factor

3 Model

We first introduce the background of Hawkes processes in Sect. 3.1 and then present the local low-rank Hawkes processes in Sects. 3.2 and 3.3. We list key notation in Table 1.

3.1 Background on Hawkes process

A temporal point process is a random process [1,4] 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 $\mathcal{T}_{t'} = \{t_i | t_i < t'\}_{i=1}^{n}$ 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:

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

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 [10], i.e.,

$$\lambda_{(u,i)}(t) = \boldsymbol{H}_{u,i} + \boldsymbol{A}_{u,i} \sum_{t_j^{u,i} \in \mathcal{T}_t^{u,i}} \kappa_\sigma \left(t - t_j^{u,i}\right), \tag{2}$$

where $\boldsymbol{H}$ denotes an $m \times n$ matrix with the (u, i)th entry equal to the nonnegative 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 [5,25], 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 [10,18] 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 [6,9,11].

3.2 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 [14]. 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 $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, \ldots, q$. 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 nonparametric 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 nonnegative 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:

$$K_h(s_1, s_2) \propto \exp\left(-\frac{1}{2}h^{-2}d(s_1, s_2)^2\right), \tag{3}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + 2 + \exp(-d(s_1, s_2)/h)}, \tag{4}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + \exp(-d(s_1, s_2)/h)}, \tag{5}$$

$$K_h(s_1, s_2) \propto \exp\left(-\frac{|d(s_1, s_2)/h|}{\sqrt{2}}\right) \cdot \sin\left(\frac{|d(s_1, s_2)/h|}{\sqrt{2}} + \frac{\pi}{4}\right). \tag{6}$$

We adopt a type of locally constant kernel regression [23] 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:

$$\lambda_s(t) = \sum_{\tau=1}^{q} \frac{K_h(s_\tau, s)}{\sum_{k=1}^{q} K_h(s_k, s)} \left[\boldsymbol{H}_s^{s_\tau} + \boldsymbol{A}_s^{s_\tau} \sum_{t_j^s \in \mathcal{T}_t^s} \kappa_\sigma\left(t - t_j^s\right) \right], \tag{7}$$

where $\boldsymbol{H}_s^{s_\tau}$ and $\boldsymbol{A}_s^{s_\tau}$ are the sth entry of the τth base intensity matrix $\boldsymbol{H}^{s_\tau}$ and self-exciting matrix $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, \ldots, q$, 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) have local low-rank structures, and the local low-rank Hawkes process in Eq. (7) is actually based on the weighted summation of q low-rank Hawkes processes in Eq. (2). Specifically, our local low-rank Hawkes model is equivalent to the low-rank Hawkes model [5] 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., $s_\tau \in 1, \ldots, q$, 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:

$$\boldsymbol{H}' = \left[\boldsymbol{H}^{s_1}, \ldots, \boldsymbol{H}^{s_q}\right], \boldsymbol{A}' = \left[\boldsymbol{A}^{s_1}, \ldots, \boldsymbol{A}^{s_q}\right], \boldsymbol{K}' = \left[K_h^{s_1}, \ldots, K_h^{s_q}\right]. \tag{8}$$

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

$$\boldsymbol{M}\{u, i\} = \left[\boldsymbol{M}^{s_1}(u, i), \ldots, \boldsymbol{M}^{s_q}(u, i)\right], \tag{9}$$

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

3.3 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 $\prod_{t_i \in \mathcal{T}} \lambda(t_i) \cdot \exp(-\int_0^T \lambda(\tau)\mathrm{d}(\tau))$, 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:

$$\mathcal{L}\left(\mathcal{T}^{u,i} \mid \boldsymbol{X}\right) = \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log\left(\boldsymbol{X}_{u,i}^\top \Phi_j^{u,i}\right) - \boldsymbol{X}_{u,i}^\top \Psi^{u,i}, \tag{10}$$

where:

$$\begin{aligned}
\boldsymbol{X}_{u,i} &= \left(\boldsymbol{H}'\{u,i\}, \boldsymbol{A}'\{u,i\}\right)^\top, \\
\Phi_j^{u,i} &= C_{u,i}^{-1} \left(\boldsymbol{K}'\{u,i\} \cdot 1, \quad \boldsymbol{K}'\{u,i\} \cdot \sum_{t_k^{u,i} < t_j^{u,i}} \kappa_\sigma\left(t_j^{u,i} - t_k^{u,i}\right) \right)^\top, \\
\Psi^{u,i} &= C_{u,i}^{-1} \left(\boldsymbol{K}'\{u,i\} \cdot T, \quad \boldsymbol{K}'\{u,i\} \cdot \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^T \kappa_\sigma\left(t - t_j^{u,i}\right) \mathrm{d}t \right)^\top.
\end{aligned} \tag{11}$$

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:

$$\mathrm{OPT} = \min_{\boldsymbol{X}} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}\left(\mathcal{T}^{u,i} \mid \boldsymbol{X}\right) + h(\boldsymbol{X}), \quad \text{s.t. } \boldsymbol{X} \geq \boldsymbol{0}, \tag{12}$$

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 [20]. Thus, minimizing $\|\boldsymbol{H}'\|_*$ and $\|\boldsymbol{A}'\|_*$ ensures each $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$ to be low-rank. After obtaining $\boldsymbol{X}$, we can use Eq. (7) to compute the intensity.

4 Parameter estimation and the algorithms

To estimate model parameters, we introduce an efficient framework to optimize the objective in Eq. (12). Specifically, we introduce the latest primal averaging conditional gradient (PA-CndG) algorithm [13] based on the Proximal Gradient (PG) method [12,19]. The algorithm, also referred to as the global approach, is described in Sect. 4.1. The convergence analysis of the algorithm is described in Sect. 4.2. We further present a parallel algorithm to increase the computation efficiency in Sect. 4.3. Finally, the computational complexity analysis is described in Sect. 4.4.

4.1 Approximate function and gradient update

Directly solving the objective in Eq. (12) is difficult because the nonnegative constraints are coupled together with the nonsmooth nuclear norm. To tackle the difficulties, we approximate

Eq. (12) by adopting a penalty method [5,24]. 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. (13):

$$\widehat{\text{OPT}} = \min_{\boldsymbol{X},\boldsymbol{Z}} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \,|\, \boldsymbol{X}) + h(\boldsymbol{Z}) + g(\boldsymbol{X}, \boldsymbol{Z}), \quad \text{s.t. } \boldsymbol{X} \geq \boldsymbol{0}, \tag{13}$$

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. (13), the nuclear norm regularization terms and the nonnegativity constraints are handled separately. The approximate objective can always be the upper bound of the real objective given the bounded ρ [5]. For simplicity, we set:

$$f(\boldsymbol{X}, \boldsymbol{Z}) = -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) + g(\boldsymbol{X}, \boldsymbol{Z}), \tag{14}$$

and the objective function becomes:

$$\widehat{\text{OPT}} = F(\boldsymbol{X}, \boldsymbol{Z}) = f(\boldsymbol{X}, \boldsymbol{Z}) + h(\boldsymbol{Z}), \quad \text{s.t. } \boldsymbol{X} \geq \boldsymbol{0}, \tag{15}$$

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

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

$$\begin{aligned} \boldsymbol{U}_1^k &= \arg\min_{\boldsymbol{U}_1^k \geq 0} \left\{ \frac{1}{2\xi_k} \left\| \boldsymbol{U}_1^k - \left(\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 f \left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1} \right) \right) \right\|^2 \right\} \\ &= \left(\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 \left(f \left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1} \right) \right) \right)_+ . \end{aligned} \tag{16}$$

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. (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 [19] and is obtained by replacing the prox-mapping with a simpler linear expansion:

$$\boldsymbol{U}_2^k = \arg\min_{\boldsymbol{Z}} \left\{ \left\langle \nabla_2 f \left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1} \right), \boldsymbol{Z} \right\rangle + h(\boldsymbol{Z}) \right\}. \tag{17}$$

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 [5,28].

4.2 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): $\gamma_k = \arg\min_{\gamma \in [0,1]} f((1-\gamma)\boldsymbol{X}^{k-1} + \gamma \boldsymbol{U}_1^k, (1-\gamma)\boldsymbol{Z}^{k-1} + \gamma \boldsymbol{U}_2^k)$ [13]. 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.

Algorithm 1: Local low-rank Hawkes

Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate ξ_k; parameters ρ, λ, β; 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_τ, b_τ) := 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

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* $\xi_k \le 1/L$. *Then we have:*

$$F\left(\boldsymbol{X}^k, \boldsymbol{Z}^k\right) - F^* \le \frac{5LD_{\max}^2}{k+1}, \tag{18}$$

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

Proof Define:

$$l_f(x, z; y_1, y_2) = f(x, z) + \langle \nabla_1 f(x, z), y_1 - x\rangle + \langle \nabla_2 f(x, z), y_2 - z\rangle. \tag{19}$$

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

$$f(y_1, y_2) \le l_f(x, z; y_1, y_2) + \frac{L}{2}\|y_1 - x\|^2 + \frac{L}{2}\|y_2 - z\|^2. \tag{20}$$

First note that:

$$\begin{aligned} \boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1} &= \gamma_k\left(\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}\right), \\ \boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1} &= \gamma_k\left(\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}\right). \end{aligned} \tag{21}$$

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

$$f\left(\boldsymbol{X}^k, \boldsymbol{Z}^k\right) \le l_f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^k, \boldsymbol{Z}^k\right) + \frac{L}{2}\left\|\boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1}\right\|^2 + \frac{L}{2}\left\|\boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1}\right\|^2$$

$$= (1-\gamma_k)\, l_f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}\right) + \gamma_k l_f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{U}_1^{k}, \boldsymbol{U}_2^{k}\right)$$
$$+ \frac{L}{2}\gamma_k^2 \left\|\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}\right\|^2 + \frac{L}{2}\gamma_k^2 \left\|\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}\right\|^2. \tag{22}$$

For simplicity, define the Bregman divergence $D(x, x') = \|x - x'\|^2$. From Eq. (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:

$$\left\langle \boldsymbol{U}_1 - \boldsymbol{Y}_1^k, \left(\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}\right)\right) - \boldsymbol{Y}_1^k \right\rangle \le 0, \quad \forall\, \boldsymbol{U}_1 \in \Omega. \tag{23}$$

Using Eq. (23) and the definition of $\boldsymbol{U}_2^k$ in Algorithm 1, we have:

$$\left\langle \nabla_1 f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}\right), \boldsymbol{Y}_1^k - \boldsymbol{U}_1^* \right\rangle \le -\frac{1}{2\xi_k}\boldsymbol{D}\left(\boldsymbol{Y}_1^k, \boldsymbol{Y}_1^{k-1}\right)$$
$$+ \frac{1}{2\xi_k}\left[\boldsymbol{D}\left(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}\right) - \boldsymbol{D}\left(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k}\right)\right] \tag{24}$$

and

$$\left\langle \nabla_2 f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}\right), \boldsymbol{U}_2^k \right\rangle + h\left(\boldsymbol{U}_2^k\right) \le \left\langle \nabla_2 f\left(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}\right), \boldsymbol{U}_2^* \right\rangle + h\left(\boldsymbol{U}_2^*\right). \tag{25}$$

Then noting that $D(x, x') \ge 0$ and using the convexity of $f(\cdot)$ and $h(\cdot)$ together with the definition of $\boldsymbol{Z}^k$ in Algorithm 1 and Eqs. (22), (24), and (25), we end up with:

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

$$+\frac{L}{2}\gamma_k\left[\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}\right)-\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k}\right)\right]$$
$$+\frac{L}{2}\gamma_k^2\boldsymbol{D}\left(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}\right)+\frac{L}{2}\gamma_k^2\boldsymbol{D}\left(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}\right). \tag{26}$$

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

$$F\left(\boldsymbol{X}^k,\boldsymbol{Z}^k\right)-F\left(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*\right)\le(1-\gamma_k)\left(F\left(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}\right)-F\left(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*\right)\right)$$
$$+\frac{L}{2}\gamma_k\left[\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}\right)-\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k})\right]$$
$$+\frac{L}{2}\gamma_k^2\boldsymbol{D}\left(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}\right)+\frac{L}{2}\gamma_k^2\boldsymbol{D}\left(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}\right). \tag{27}$$

In view of Lemma 1 of [13] 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:

$$F\left(\boldsymbol{X}^k,\boldsymbol{Z}^k\right)-F\left(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*\right)\le\Gamma_k(1-\gamma_1)\left(F\left(\boldsymbol{X}^0,\boldsymbol{Z}^0\right)-F\left(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*\right)\right)$$
$$+\frac{\Gamma_k L}{2}\sum_{i=1}^{k}\left\{\frac{\gamma_i}{\Gamma_i}\left[\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{i-1}\right)-\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{i}\right)\right]\right.$$
$$\left.+\frac{\gamma_i^2}{\Gamma_i}\left[\boldsymbol{D}\left(\boldsymbol{U}_1^i,\boldsymbol{U}_1^{i-1}\right)+\boldsymbol{D}\left(\boldsymbol{U}_2^i,\boldsymbol{U}_2^{i-1}\right)\right]\right\}. \tag{28}$$

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

$$F\left(\boldsymbol{X}^k,\boldsymbol{Z}^k\right)-F^*\le\frac{L}{k(k+1)}\left\{k\boldsymbol{D}\left(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^0\right)+2\sum_{i=1}^{k}\left[\boldsymbol{D}\left(\boldsymbol{U}_1^i,\boldsymbol{U}_1^{i-1}\right)+\boldsymbol{D}\left(\boldsymbol{U}_2^i,\boldsymbol{U}_2^{i-1}\right)\right]\right\}\le\frac{5LD_{\max}^2}{k+1}. \tag{29}$$

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

4.3 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. (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:

$$\mathcal{L}^{s_\tau}(\mathcal{T}^{u,i}\mid\boldsymbol{X}^{s_\tau})=\frac{1}{q}\left\{\sum_{t_j^{u,i}\in\mathcal{T}^{u,i}}\log\left(\boldsymbol{X}(s_\tau)_{u,i}^\top\Phi(s_\tau)_j^{u,i}\right)-\boldsymbol{X}(s_\tau)_{u,i}^\top\Psi(s_\tau)^{u,i}\right\}, \tag{30}$$

where:

$$\begin{aligned}
\boldsymbol{X}(s_\tau)_{u,i} &= \left(\boldsymbol{H}^{s_\tau}(u,i), \boldsymbol{A}^{s_\tau}(u,i)\right)^\top, \\
\Phi(s_\tau)_j^{u,i} &= q\left(1, \sum_{t_k^{u,i} < t_j^{u,i}} \kappa_\sigma\left(t_j^{u,i} - t_k^{u,i}\right)\right)^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}, \\
\Psi(s_\tau)^{u,i} &= q\left(T, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{T} \kappa_\sigma\left(t - t_j^{u,i}\right) \mathrm{d}t\right)^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}.
\end{aligned} \tag{31}$$

Then we define the parallel objective function as:

$$\mathrm{OPT}_p = \min_{\boldsymbol{X}^{s_\tau}, \boldsymbol{Z}^{s_\tau}} -\frac{1}{|\mathcal{O}|} \sum_{\tau=1}^{q} \left\{ \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \mathcal{L}^{s_\tau}\left(\mathcal{T}^{u,i} | \boldsymbol{X}^{s_\tau}\right) + h_{s_\tau}(\boldsymbol{Z}^{s_\tau}) \right\}, \quad \text{s.t. } \boldsymbol{X} \geq \boldsymbol{0}, \tag{32}$$

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

Algorithm 2: Local low-rank Hawkes parallel

Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate ξ_k; parameters ρ, λ, β; 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 $\tau = 1, \ldots, q$ ***in parallel*** **do**
  (a_τ, b_τ) := 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

Theorem 2 *With the condition that* λ^τ *and* β^τ *for* $\tau = 1, \ldots, q$ *satisfy Eq. (33), the optimal value* OPT_p *in Eq. (32) coincides with the global optimal value OPT in Eq. (15).*

$$\lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_* \leq \sum_{\tau=1}^{q} \left(\lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*\right). \tag{33}$$

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

$$\varphi\left(\frac{\sum \alpha_i x_i}{\alpha_i}\right) \leq \frac{\sum \alpha_i \varphi(x_i)}{\sum \alpha_i}. \tag{34}$$

The equality holds if and only if $x_1 = x_2 = \cdots = x_n$ or $\varphi(\cdot)$ is linear. Specifically, Eq. (34) becomes:

$$\varphi\left(\frac{\sum x_i}{n}\right) \leq \frac{\sum \varphi(x_i)}{n} \tag{35}$$

if the weighs α_i are equal.

As $-\log(\cdot)$ is convex, we rewrite Eq. (10) based on Eq. (35) as:

$$\begin{aligned}
-\mathcal{L}\left(\mathcal{T}^{u,i} \mid \{\boldsymbol{X}^{s_\tau}\}_{\tau=1}^{q}\right) &= -\sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log\left(\sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}/q\right) + \sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i}/q \\
&\leq -\sum_{\tau=1}^{q} \left\{ \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log\left(\boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}\right) - \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i} \right\} /q \\
&= -\sum_{\tau=1}^{q} \mathcal{L}^{s_\tau}\left(\mathcal{T}^{u,i} \mid \boldsymbol{X}^{s_\tau}\right).
\end{aligned} \tag{36}$$

Given Eq. (33), we have:

$$h(\boldsymbol{Z}) = \lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_* \leq \sum_{\tau=1}^{q} \left(\lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*\right) = \sum_{\tau=1}^{q} h_{s_\tau}\left(\boldsymbol{Z}^{s_\tau}\right). \tag{37}$$

Therefore, plugging Eqs. (36) and (37) into the previous objective function in Eq. (12), we have $\mathrm{OPT} \leq \mathrm{OPT}_p$ and readily arrive at the theorem. □

Therefore, we can optimize the parallel objective function in Eq. (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.

4.4 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 $O(N^2 q C/\epsilon)$, where $N = \max\{m, n\}$, since we have $2 \times m \times n \times q$ entries in the global model parameter

Table 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

matrix $\boldsymbol{X}$ 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. 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 $O(PC/\epsilon)$.

5 Distance measure, kernel calculation, and anchor point selection

The distance metric d such as Eq. (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 [14] to factorize the matrix using standard incomplete SVD [2] to obtain the latent matrices. We also investigate different combination methods in Sect. 6.3.2 to construct a unified feature based on both user–item interactions and content features, and compare the performance.

In Sects. 3.2 and 4, 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 [14], 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. (10), we use the Gaussian kernel in Eq. (3) for both K and K', as the kernel function is nonzero 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 $\tau = 1, \ldots, q$. 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 Sect. 6.3.3. We also compare the performance of kernel smoothing with different numbers of anchor points in Sect. 6.3.4.

There are several approaches for selecting the anchor points $s_1, \ldots, s_q$, 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 Sect. 6.3.5.

6 Experiments

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

6.1 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.

IPTV dataset [26] 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 nearly 2.4M events and 1420 movie features such as genres. These features are only used for Coevolve baseline and experiments in Sect. 6.3.2 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 $S_{(u,i)}(t) = \exp(-\int_{t_n^{u,i}}^{t} \lambda_{(u,i)}(\tau)\mathrm{d}(\tau))$. 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 [24], 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) [5,24] 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. (13) with fixed coefficients of Hawkes processes learned using events in the training set.

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

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

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

Poisson tensor 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 [3]. 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.

Low-rank Hawkes is a Hawkes process-based model [5] 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 [24] 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.

6.3 Results

In Sect. 6.3.1, 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 Sects. 6.3.2, 6.3.3, 6.3.4, and 6.3.5, respectively. Finally, we compare the model performance of the global and parallel algorithms in Sect. 6.3.6.

6.3.1 Comparison with baselines

As described in Sect. 5, we follow [14] to factorize the user–item interaction matrix using standard incomplete SVD [2] 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. 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. *Low-rank Hawkes* treats each (u, i) pair's process independently. *Poisson* and *Poisson tensor* 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 [24] because the individual preferences are influenced by the general preference. Meanwhile, we can see

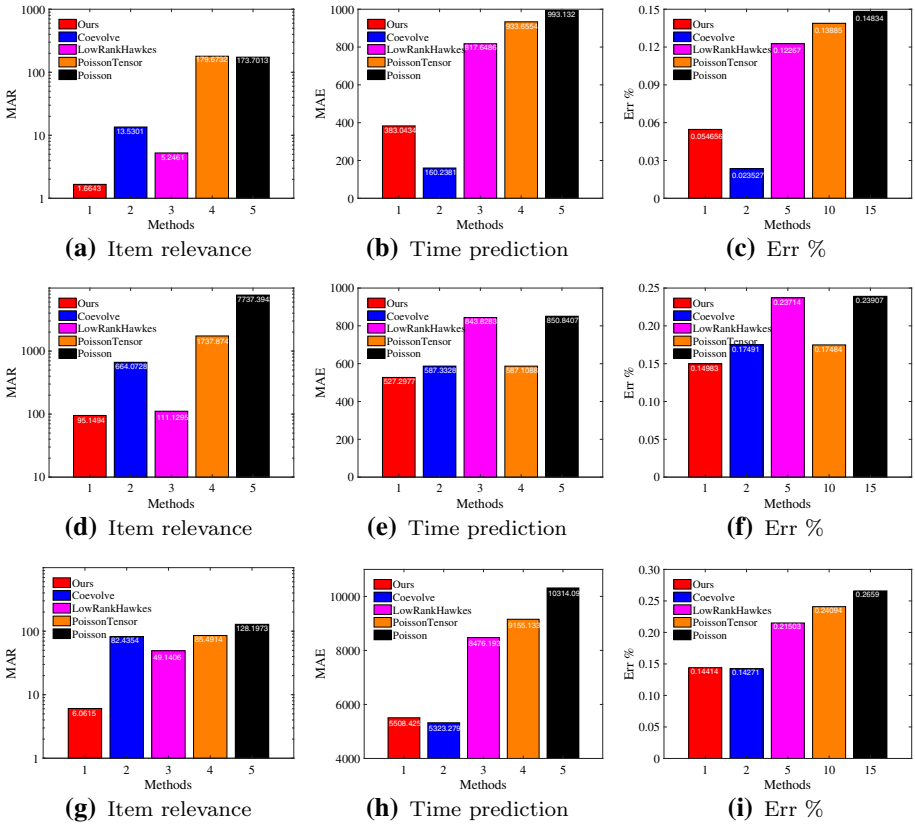


Fig. 1 Prediction accuracy on IPTV (top row), Yelp (middle row), and Reddit (bottom row)

that the Hawkes process-based models, such as our model, *Coevolve*, and *Low-rank Hawkes*, 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 *Low-rank Hawkes* and *Poisson*, which means that the history is not such an important factor. In this time sparsity case, factorization model *Poisson tensor* 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 some interaction influences from similar groups. In other words, our model performs better on sequences without sufficient events.

6.3.2 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)

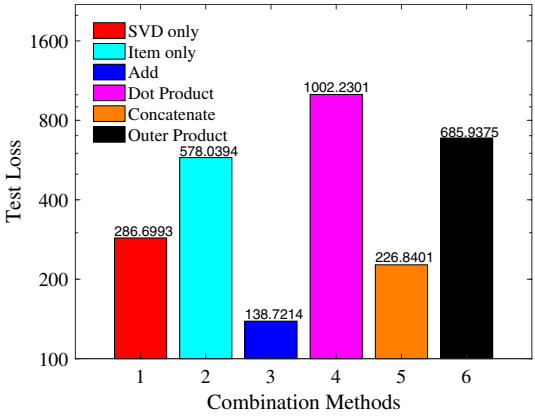


Fig. 2 Test loss with respect to different combination methods of item features on IPTV dataset

(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.

Figure 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 are 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 principal component analysis (PCA), autoencoder (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.

6.3.3 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. 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.

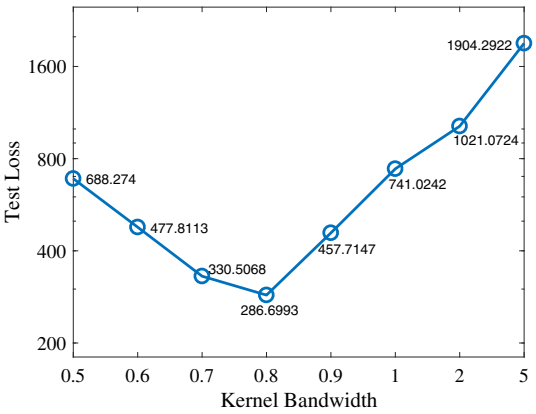


Fig. 3 Test loss versus smoothing kernel bandwidth on IPTV dataset

Table 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**

In each row, the bold value indicates the best result

6.3.4 Effect of the number of anchor points

We also compare the performance of kernel smoothing with different numbers of anchor points in Table 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 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 (5–10). 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. (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.

6.3.5 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 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 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 pair-wise distance between pairs of points and different metrics for computing the distance between clusters. As shown in Table 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-wise distances. For hierarchical clustering, we adopt the best strategy based on the cophenetic correlation. As shown in Table 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 intro-

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

[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 4 Cophenetic 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 bold values indicates the best result

Table 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

In each row, the bold value indicates the best result

duces additional computational cost (e.g., $O(P^2)$ complexity for K-means clustering and $O(P^3)$ for hierarchical clustering) in the preprocessing. Selecting sufficiently large number of anchor points randomly may achieve comparable performance but increase model complexity as described in Sect. 4.4. In cases where the clustering complexity is negligible comparing to model complexity, a well-chosen hierarchical clustering strategy is preferred.

6.3.6 Comparison of global and parallel algorithms

We also compare the results of the global and parallel algorithms and present the results in Table 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 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 $(\lambda_\tau,\beta_\tau,\rho_\tau)$ 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.

Table 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**

In each row, the bold value indicates the best result

7 Conclusions

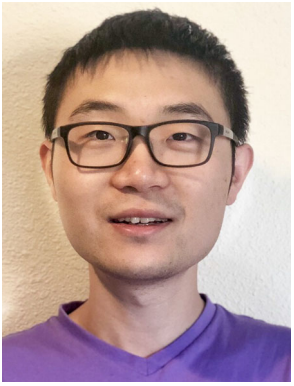
In this paper, we present a novel framework that integrates the kernel smoothing and the Hawkes process to model the temporal events of user–item interactions. We assume that the intensity parameter matrix is locally low-rank. With nonparametric kernel smoothing, each user–item pair can be simulated 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. Our 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. Further work includes extending to other application areas and integrating the framework with certain deep neural network structures.

Acknowledgements This work was supported in part by the Louisiana Board of Regents under Grant LEQSF(2017-20)-RD-A-29. The authors would also like to thank Yichen Wang and Le Song from Georgia Tech for their helpful discussions.

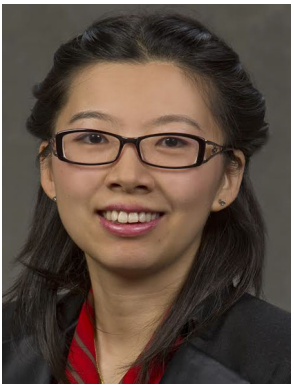
References

1. Aalen O, Borgan O, Gjessing H (2008) Survival and event history analysis: a process point of view. Springer, New York
2. Bell R, Koren Y, Volinsky C (2007) Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 95–104
3. Chi EC, Kolda TG (2012) On tensors, sparsity, and nonnegative factorizations. SIAM J Matrix Anal Appl 33(4):1272–1299
4. Cox DR, Lewis PAW (1972) Multivariate point processes. In: Proceedings of the 6th Berkeley symposium on mathematical statistics and probability, vol 3, pp 401–448
5. Du N, Wang Y, He N, Sun J, Song L (2015) Time-sensitive recommendation from recurrent user activities. In: Proceedings of the annual conference on neural information processing systems (NIPS), pp 3492–3500
6. Eichler M, Dahlhaus R, Dueck J (2017) Graphical modeling for multivariate Hawkes processes with nonparametric link functions. J Time Ser Anal 38(2):225–242
7. Etesami J, Kiyavash N, Zhang K, Singhal K (2016) Learning network of multivariate Hawkes processes: a time series approach. In: Proceedings of the conference on uncertainty in artificial intelligence (UAI), pp 162–171

8. Farajtabar M, Du N, Rodriguez MG, Valera I, Zha H, Song L (2014) Shaping social activity by incentivizing users. In: Proceedings of the annual conference on neural information processing systems (NIPS), pp 2474–2482
9. Hall EC, Willett RM (2016) Tracking dynamic point processes on networks. IEEE Trans Inf Theory 62(7):4327–4346
10. Hawkes AG (1971) Spectra of some self-exciting and mutually exciting point processes. Biometrika 58(1):83–90
11. Krumin M, Reutsky I, Shoham S (2010) Correlation-based analysis and generation of multiple spike trains using Hawkes models with an exogenous input. Front Comput Neurosci 4:147
12. Lan G (2012) An optimal method for stochastic composite optimization. Math Program 133(1):365–397
13. Lan G (2013) The complexity of large-scale convex programming under a linear optimization oracle. arXiv:1309.5550
14. Lee J, Kim S, Lebanon G, Singer Y (2013) Local low-rank matrix approximation. In: Proceedings of the international conference on machine learning (ICML), pp 82–90
15. Lee J, Sun M, Kim S, Lebanon G (2012a) Automatic feature induction for stagewise collaborative filtering. In: Proceedings of the annual conference on neural information processing systems (NIPS), pp 314–322
16. Lee J, Sun M, Lebanon G (2012b) PREA: personalized recommendation algorithms toolkit. J Mach Learn Res 13(1):2699–2703
17. Lemonnier R, Scaman K, Kalogeratos A (2017) Multivariate Hawkes processes for large-scale inference. In: Proceedings of the AAAI conference on artificial intelligence, pp 2168–2174
18. Liniger TJ (2009) Multivariate Hawkes processes. PhD thesis, ETH Zurich
19. Nesterov Y (2013) Gradient methods for minimizing composite functions. Math Program 140(1):125–161
20. Sastry S (1990) Some NP-complete problems in linear algebra. Honors projects
21. Shang J, Sun M (2018) Local low-rank Hawkes processes for temporal user–item interactions. In: Proceedings of the IEEE international conference on data mining (ICDM), IEEE
22. Sun M, Lebanon G, Kidwell P (2011) Estimating probabilities in recommendation systems. In: Proceedings of the international conference on artificial intelligence and statistics (AISTATS), pp 734–742
23. Wand MP, Jones MC (1995) Kernel smoothing. Chapman and Hall/CRC, Boca Raton
24. Wang Y, Du N, Trivedi R, Song L (2016) Coevolutionary latent feature processes for continuous-time user–item interactions. In: Proceedings of the annual conference on neural information processing systems (NIPS), pp 4547–4555
25. Xiao W, Xu X, Liang K, Mao J, Wang J (2016) Job recommendation with Hawkes process: an effective solution for RecSys challenge 2016. In: Proceedings of the recommender systems challenge
26. Xu H, Farajtabar M, Zha H (2016) Learning granger causality for Hawkes processes. In: Proceedings of the international conference on machine learning (ICML), pp 1717–1726
27. Xu H, Wu W, Nemati S, Zha H (2017) Patient flow prediction via discriminative learning of mutually-correcting processes. IEEE Trans Knowl Data Eng 29(1):157–171
28. Yu AW, Ma W, Yu Y, Carbonell J, Sra S (2014) Efficient structured matrix rank minimization. In: Proceedings of the annual conference on neural information processing systems (NIPS), pp 1350–1358
29. Zhou K, Zha H, Song L (2013) Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In: Proceedings of the international conference on artificial intelligence and statistics (AISTATS), pp 641–649



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.



Mingxuan Sun received the B.S. degree in computer science and engineering from Zhejiang University, Hangzhou, China in 2004, the M.S. degree in computer science from University of Kentucky, Lexington, KY, USA in 2006, and the Ph.D. degree in computer science from Georgia Institute of Technology, Atlanta, GA, USA in 2012. Since August 2015, she has been an Assistant Professor with the Division of Computer Science and Engineering, Louisiana State University, Baton Rouge, LA, USA. Prior to that, she was a Senior Scientist with Pandora Media, Inc., Oakland, CA, USA, from 2012 to 2015. Her research interests include machine learning, deep learning, and information retrieval. She is also interested in machine learning applications in information security and wireless communications.