A Plug-and-Play Bregman ADMM Module for Inferring Event Branches in Temporal Point Processes

Qingmei Wang[1], Yuxin Wu[1], Yujie Long[2], Jing Huang[2], Fengyuan Ran[2], Bing Su[1], Hongteng Xu[1]*

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Cyber Science and Engineering, Wuhan University

Abstract

An event sequence generated by a temporal point process is often associated with a hidden and structured event branching process that captures the triggering relations between its historical and current events. In this study, we design a new plug-and-play module based on the Bregman ADMM (BADMM) algorithm, which infers event branches associated with event sequences in the maximum likelihood estimation framework of temporal point processes (TPPs). Specifically, we formulate the inference of event branches as an optimization problem for the event transition matrix under sparse and low-rank constraints, which is embedded in existing TPP models or their learning paradigms. We can implement this optimization problem based on subspace clustering and sparse group-lasso, respectively, and solve it using the Bregman ADMM algorithm, whose unrolling leads to the proposed BADMM module. When learning a classic TPP (e.g., Hawkes process) by the expectation-maximization algorithm, the BADMM module helps derive structured responsibility matrices in the E-step. Similarly, the BADMM module helps derive low-rank and sparse attention maps for the neural TPPs with self-attention layers. The structured responsibility matrices and attention maps, which work as learned event transition matrices, indicate event branches, e.g., inferring isolated events and those key events triggering many subsequent events. Experiments on both synthetic and real-world data show that plugging our BADMM module into existing TPP models and learning paradigms can improve model performance and provide us with interpretable structured event branches.

Code —
https://github.com/qingmeiwangdaily/BADMM_TPP

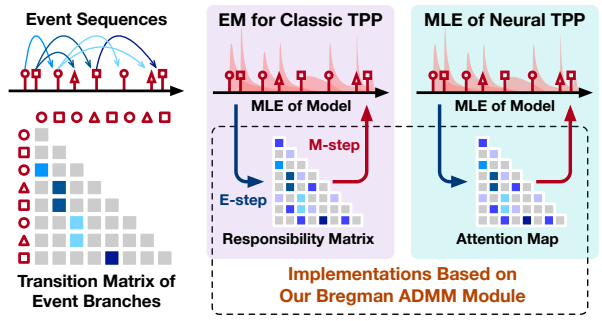
Introduction

Temporal point processes (TPPs) are powerful tools for modeling events that occur sequentially in continuous-time domain (Kingman 1992; Ross et al. 1996). They have been extensively used to capture the dynamics of the events in various domains, such as earthquake prediction (Lewis and Shedler 1979), financial analysis (Bacry, Mastromatteo, and Muzy 2015), social network modeling (Zhou, Zha, and Song 2013a), recommendation systems (Xu, Carin, and Zha 2018), and so on. Typically, for an event sequence generated by a TPP, there exists a branching process capturing the event-level triggering patterns hidden in the sequence (Dion and Yanev 1994; Møller and Rasmussen 2006). As illustrated in Figure 1(a), the branching process is often represented as a transition matrix of events, which provides insights into the causal relationships between events and helps identify the cascades of subsequent events. Therefore, inferring such event branches from observed event sequences is crucial for us to reveal the underlying mechanism of event generation and information diffusion, e.g., identifying the source node in an information diffusion network (Farajtabar et al. 2015; Zhang et al. 2018) and controlling the diffusion of specific information (Farajtabar et al. 2014, 2017a).

Despite its significance, inferring event branches is challenging because they are latent variables unobservable in practice. For some classic TPPs like Hawkes process (Zhou, Zha, and Song 2013b), we can learn the TPPs based on observed event sequences in the expectation-maximization (EM) framework, in which the event branches are often inferred as the responsibilities of the events (i.e., the posterior probabilities of historical events given the current ones). For neural TPPs like transformer Hawkes process (THP) (Zuo et al. 2020) and self-attentive Hawkes process (SAHP) (Zhang et al. 2020), the attention maps within the models indicate the impacts of historical events on the current ones, and thus, can be treated as evidence of event branches. However, both methods often suffer the over-smoothness issue and lead to unstructured event branches. How to infer interpretable and structured event branches effectively for various TPP models is still an open problem.

In this study, we introduce a novel plug-and-play module based on the Bregman Alternating Direction Method of Multipliers (Bregman ADMM or BADMM for short) algorithm (Wang and Banerjee 2014), which helps infer event branches in the maximum likelihood estimation framework of TPPs. As illustrated in Figure 1(b), given the responsibility matrices of classic TPPs or the attention maps of neural TPPs, the BADMM module imposes low-rank and sparse structures on the matrices by solving a subspace clustering problem (Elhamifar and Vidal 2013) or a sparse group-lasso problem (Simon et al. 2013). Applying the BADMM module to the EM algorithm of classic TPPs, we can optimize the responsibility matrices iteratively to guide the learning of

*Corresponding author: hongtengxu@ruc.edu.cn



(a) Event branches (b) Plug-in strategies of BADMM module

Figure 1: (a) An illustration of event branches and the corresponding transition matrix. (b) Plug-in strategies of BADMM module for classic and neural TPPs.

model parameters. For neural TPPs, we unroll the iterations of the Bregman ADMM algorithm and build the BADMM module to obtain a new attention layer. With the help of the BADMM module, we effectively penalize dense triggering patterns among events for both classic and neural TPPs, leading to structured event transition matrices and, accordingly, interpretable event branches.

Through extensive experiments on both synthetic and real-world datasets, we evaluate different implementations of the BADMM module and demonstrate their effectiveness. Experimental results show that incorporating the BADMM module into existing TPP models improves their performance and interpretability. In particular, the BADMM module not only enhances the predictive accuracy of TPP models but also provides valuable insights into the hidden event branching processes, e.g., identifying the isolated events within event sequences and those key events triggering multiple subsequent events.

Related Work and Preliminaries

Temporal Point Processes

Temporal point process is a classic statistical tool to model the event sequences in continuous-time domain (Kingman 1992; Wang et al. 2023). Suppose that we observe an event sequence $\boldsymbol{s} = \{(t_n, c_n)\}_{n=1}^{N}$, where (t_n, c_n) is the n-th event, $t_n \in [0, T]$ is its timestamp, and $c_n \in \mathcal{C} = \{1, ..., C\}$ is its event type. A TPP often models the dynamics of the events by a parametric multivariate intensity function, denoted as $\boldsymbol{\lambda}(t;\theta) = \{\lambda_c(t;\theta)\}_{c\in\mathcal{C}, t\in[0,T]}$, where

$$\lambda_c(t;\theta)\mathrm{d}t = \mathrm{d}\mathbb{E}[N_c(t;\theta)|\mathcal{H}_t^{\mathcal{C}}],\ \forall c \in \mathcal{C}, \tag{1}$$

represents the expected instantaneous rate of the type-c event happening at time t given historical events. Here, $N_c(t;\theta)$ is the counting process of the event type, which records the number of the type-c events happening till time t, and $\mathcal{H}_t^{\mathcal{C}} = \{(t_n, c_n) \in \boldsymbol{s} | t_n < t\}$ is the history till time t. The parameters of the intensity function (and the counting process) are denoted as θ. We can learn the TPP in the maximum likelihood estimation (MLE) framework, maximizing the likelihood of the event sequence shown below:

$$L(\boldsymbol{s};\theta) = \prod_{n=1}^{N} \lambda_{c_n}(t_n;\theta) \exp\Big(-\sum_{c\in\mathcal{C}} \int_0^T \lambda_c(s;\theta)\mathrm{d}s\Big). \tag{2}$$

Classic TPP models, like Poisson process (Kingman 1992), Hawkes process (Hawkes 1971), self-correcting process (Isham and Westcott 1979), and their multivariate versions (Zhou, Zha, and Song 2013b; Xu et al. 2016), often apply manually designed intensity functions, which simplifies the learning problem but suffers a high risk of model misspecification. To overcome this challenge, some neural TPPs have been proposed to model intensity functions by neural networks, which can enhance model capabilities significantly. The early neural TPPs are built based on recurrent neural networks, e.g., RMTPP (Du et al. 2016) and NHP (Mei and Eisner 2017). Recently, some Transformer-based TPP models are proposed, e.g., SAHP (Zhang et al. 2020) and THP (Zuo et al. 2020).

Inference of Event Branches

For the event sequence whose events have inter-dependency, there always exists a branching process associated with the corresponding TPP (Møller and Rasmussen 2006; Farajtabar et al. 2014). Typically, we can represent the branches of N events by a lower triangular matrix, denoted as $\boldsymbol{B} = [b_{nn'}] \in [0, \infty)^{N\times N}$, where $b_{nn'} \geq 0$ if $n \geq n'$ and $b_{nn'} = 0$ otherwise. This matrix often works as a "transition" matrix of events — the nonzero $b_{nn'}$ indicates that the n'-th event contributes to the happening of the n-th event. Ideally, a sparse and structured transition matrix helps reveal the underlying generative mechanisms of events (Dion and Yanev 1994; González et al. 2013; Champion et al. 2022), which enhances the interpretability of the TPP model. Therefore, in this study, we aim to infer the event branch matrix $\boldsymbol{B}$ simultaneously when learning the TPP model θ.

Inferring event branches for arbitrary TPPs is challenging due to the latent nature of transition matrix. Focusing on Hawkes process (Zhou, Zha, and Song 2013b), whose intensity function encodes the branching process explicitly (Møller and Rasmussen 2006), some attempts have been made to infer event branches. For example, an information source identification method is proposed in (Farajtabar et al. 2015), which traces back to information sources in diffusion networks from partially observed cascades, crucial for applications like epidemiology and social networks. A related study (Farajtabar et al. 2017b) designs interventions to mitigate fake news spread using point process models, demonstrating the real-world application of branch process inference in misinformation control. Focusing on Hawkes processes with textual covariates, the work in (Zhang et al. 2018) identifies root sources in textual conversation threads, providing insights into propagation patterns within text data. However, the above methods often lead to non-structured and over-dense event branches because they seldom consider imposing structural regularization on the transition matrix of events. Moreover, these methods only apply to Hawkes process, which cannot be extended to other complicated TPP models.

Optimization-driven Model Design

In this study, we aim to design a plug-and-play BADMM module that effectively infers event branches for both classic and neural TPPs. Our work can be treated as a new attempt at optimization-driven model design. In this field, some methods have been made to implement implicit neural network layers based on optimization algorithms. For example, the Sinkhormer in (Sander et al. 2022) and the ROTP in (Xu and Cheng 2023) apply Sinkhorn-scaling algorithm (Cuturi 2013) to improve attention maps and global pooling layers, respectively. An ADMM-based neural network is proposed to solve compressive sensing problems (Yang et al. 2018). These methods can embed optimization algorithms into model architectures or learning paradigms and, accordingly, impose structural constraints on model parameters or intermediate outputs. However, none of the existing methods consider inferring event branches for TPPs.

Proposed Method

Transition Matrices within TPPs

As aforementioned, the inference of event branches corresponds to learning structured transition matrix of events. This matrix is intrinsic for many existing TPP models.

Hawkes process The intensity function of Hawkes process encodes event branches explicitly (Møller and Rasmussen 2006), which is constructed as

$$\lambda_c(t;\theta) = \mu_c + \sum\nolimits_{t_n<t} a_{cc_n}\,\kappa(t-t_n), \tag{3}$$

in which μ_c is time-invariant exogenous intensity of the type-c event and $\sum_{t_n<t} a_{cc_n}\kappa(t-t_n)$ is endogenous intensity recording the accumulative impacts of historical events at time t. $\kappa(t)$, $t \geq 0$, is a predefined time-decay function. As a result, the model parameters $\theta = \{\boldsymbol{\mu}, \boldsymbol{A}\}$ include the exogenous intensity vector $\boldsymbol{\mu} = [\mu_c] \in [0,\infty)^C$ and the infectivity matrix $\boldsymbol{A} = [a_{cc'}] \in [0,\infty)^{C\times C}$ capturing the pairwise impacts between different event types.

We often apply the EM algorithm (Zhou, Zha, and Song 2013b) to achieve the MLE of Hawkes process. In the t-th iteration, given the current parameter $\theta^{(t)} = \{\boldsymbol{\mu}^{(t)}, \boldsymbol{A}^{(t)}\}$, we update the parameter by the following two steps:

E-step: We construct a lower bound of the log-likelihood in (2) based on Jensen's inequality:

$$\begin{aligned}
&\log L(\boldsymbol{s};\theta) \geq Q(\theta,\theta^{(t)}) \\
&= \sum_{n=1}^{N}\Big(r_{nn}^{(t)}\log\frac{\mu_{c_n}}{r_{nn}^{(t)}} + \sum_{n'=1}^{n-1} r_{nn'}^{(t)}\log\frac{a_{c_nc_{n'}}\kappa(t_n-t_{n'})}{r_{nn'}^{(t)}}\Big) \\
&\quad - \sum\nolimits_{c\in\mathcal{C}}\Big(T\mu_c + \sum\nolimits_{n=1}^{N} a_{cc_n}\int_0^{T-t_n}\kappa(s)\mathrm{d}s\Big),
\end{aligned}$$

where $\boldsymbol{R}^{(t)} = [r_{nn'}^{(t)}]$ is the responsibility matrix in the t-th iteration. Its element is

$$r_{nn'}^{(t)} = \begin{cases} \frac{\mu_{c_n}^{(t)}}{\lambda_{c_n}(t_n;\theta^{(t)})}, & n = n', \\ \frac{a_{c_nc_{n'}}^{(t)}\kappa(t_n-t_{n'})}{\lambda_{c_n}(t_n;\theta^{(t)})}, & n > n', \\ 0, & n < n', \end{cases} \tag{4}$$

which corresponds to the posterior probability of the n'-th event given the n-th event.

M-step: We update the model parameter by minimizing the surrogate function $Q(\theta,\theta^{(t)})$, i.e.,

$$\theta^{(t+1)} = \arg\max_\theta Q(\theta,\theta^{(t)}), \tag{5}$$

which can be solved in a closed form (Zhou, Zha, and Song 2013b; Xu, Farajtabar, and Zha 2016).

We can find that the responsibility matrix in (4) works as the transition matrix of events, which is estimated by the E-step iteratively. Therefore, the inference of event branches corresponds to estimating a structured responsibility matrix.

Transformer-based neural TPPs For Transformer-based neural TPPs (Zuo et al. 2020; Zhang et al. 2020), their intensity functions encode historical impacts by a self-attention mechanism. Given the event embeddings of a sequence, denoted as $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$, the attention map is

$$\tilde{\boldsymbol{A}} = \sigma_r\Big(\boldsymbol{L}\odot\frac{(\boldsymbol{Q}\boldsymbol{X})^\top\boldsymbol{K}\boldsymbol{X}}{\sqrt{D}}\Big), \tag{6}$$

where $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{d\times D}$ maps the event embeddings to latent spaces. $\boldsymbol{L} = [\ell_{nn'}]$ is a lower triangular masking matrix, whose element $\ell_{nn'} = 1$ if $n \geq n'$, otherwise, $\ell_{nn'} = \infty$. "$\odot$" denotes the Hadamard product of matrix, and σ_r means applying the Softmax operation to each row of input matrix. As shown in (Zuo et al. 2020), the sparsity of $\tilde{\boldsymbol{A}}$ indicates event branches — $a_{nn'}$ with a large value implies that the n-th event is likely to be triggered by the n'-th event.

Over-smoothness issue The above two examples demonstrate that many TPP models estimate transition matrices intrinsically either in their learning paradigms or their model architectures. Unfortunately, these matrices often suffer the over-smoothness issue, i.e., having too many nonzero elements and leading to over-dense and unstructured event branches. For the $\boldsymbol{R}^{(t)}$ in (4), its structure is determined by the sparsity of $\boldsymbol{\mu}$ and $\boldsymbol{A}$, which requires additional sparse regularization (Xu, Farajtabar, and Zha 2016; Zhou, Zha, and Song 2013b). For the $\tilde{\boldsymbol{A}}$ in (6), its lower-triangular part is always nonzero because of using the softmax operation. Therefore, we need to impose more structural constrains when estimating the transition matrices, which motivates the design of our BADMM module.

BADMM Module for Structured Event Branches

Imposing sparse and low-rank structures Taking the $\boldsymbol{R}^{(t)}$ in (4) or the $\tilde{\boldsymbol{A}}$ in (6) as the initial transition matrix, denoted as $\boldsymbol{B}_0$, our Bregman ADMM module imposes sparse and low-rank structures on the matrix by solving the following optimization problem:

$$\min_{\boldsymbol{B}\in\Omega}\underbrace{\mathrm{KL}(\boldsymbol{B}\parallel\boldsymbol{B}_0)}_{\text{Prior}} + \lambda\underbrace{(\alpha\|\boldsymbol{B}\|_1 + (1-\alpha)R(\boldsymbol{B}))}_{\text{Structural regularizer}}. \tag{7}$$

The feasible domain $\Omega = \{\boldsymbol{B} = [b_{nn'}] | \boldsymbol{B}\mathbf{1}_N = \mathbf{1}_N, b_{nn'} = 0 \text{ for } n < n',\ b_{nn'} \geq 0 \text{ for } n \geq n'\}$, ensuring $\boldsymbol{B}$ is a row-normalized lower triangular matrix. In the objective function, the first term penalizes the KL-divergence between $\boldsymbol{B}$

and the initial $\boldsymbol{B}_0$, which requires the target transition matrix inherits the prior knowledge provided by $\boldsymbol{B}_0$ to some extent. The second term is a joint sparse and low-rank regularization, in which the ℓ_1-norm $\|\boldsymbol{B}\|_1 = \sum_{n,n'=1}^{N} |b_{nn'}|$ enhances the sparsity of the event branch matrix and $R(\boldsymbol{B})$ penalizes the rank of $\boldsymbol{B}$. In this study, we consider two implementations of $R(\boldsymbol{B})$:

- **Subspace clustering.** We can implement $R(\boldsymbol{B})$ as the nuclear norm, i.e., $\|\boldsymbol{B}\|_* = \sum_{n=1}^{N} \sigma_N(\boldsymbol{B})$. It computes the summation of $\boldsymbol{B}$'s singular values, which encourages sparse singular values and results in a low-rank matrix. As a result, the regularizer $\alpha\|\boldsymbol{B}\|_1 + (1-\alpha)\|\boldsymbol{B}\|_*$ corresponds to the subspace clustering method in (Elhamifar and Vidal 2013; Liu, Lin, and Yu 2010).
- **Sparse group-lasso.** We can also implement $R(\boldsymbol{B})$ as the $\ell_{1,2}$ norm, i.e., $\|\boldsymbol{B}\|_{1,2} = \sum_{n=1}^{N} \|\boldsymbol{b}_n\|_2$, where $\boldsymbol{b}_n$ is the n-th column of $\boldsymbol{B}$. This term encourages $\boldsymbol{B}$ to have a group sparse structure. As a result, the regularizer $\alpha\|\boldsymbol{B}\|_1 + (1-\alpha)\|\boldsymbol{B}\|_*$ corresponds to the sparse group-lasso in (Simon et al. 2013).

The significance of the regularization term is controlled by $\lambda > 0$, and the trade-off between the sparse and low-rank terms is achieved by $\alpha \in [0, 1]$.

Remark. It should be noted that the utilization of the regularization is motivated by the event branching structure we desire. As illustrated in Figure 1(a), an event branching process often consists of some isolated events independent of other events and some key events that trigger subsequent events. In addition, an event's impact always decays over time, so it can only trigger a subset rather than all subsequent events. As a result, the corresponding event branch matrix is sparse and low-rank, and the proposed regularizer enhances such structures accordingly.

Implementation details Applying Bregman ADMM algorithm (Wang and Banerjee 2014), we can solve the above optimization problem iteratively. In particular, we first rewrite (7) in the following equivalent format:

$$\begin{aligned} &\min_{\boldsymbol{B},\boldsymbol{X}_1,\boldsymbol{X}_2} \mathrm{KL}(\boldsymbol{B}\|\boldsymbol{B}_0) + \lambda(\alpha\|\boldsymbol{X}_1\|_1 + (1-\alpha)R(\boldsymbol{X}_2)) \\ &\text{s.t.} \quad \boldsymbol{B} = \boldsymbol{X}_1 = \boldsymbol{X}_2, \quad \boldsymbol{B} \in \Omega. \end{aligned} \tag{8}$$

Here, $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are two auxiliary variables helping decouple the three terms in the objective function. Then, we can further rewrite (8) in its augmented Lagrangian form, i.e.,

$$\begin{aligned} &\max_{\boldsymbol{Z}_1,\boldsymbol{Z}_2} \min_{\boldsymbol{B}\in\Omega,\boldsymbol{X}_1,\boldsymbol{X}_2} \mathrm{KL}(\boldsymbol{B} \parallel \boldsymbol{B}_0) \\ &\quad + \lambda(\alpha\|\boldsymbol{X}_1\|_1 + (1-\alpha)R(\boldsymbol{X}_2)) \\ &\quad + \rho\sum\nolimits_{i=1,2}\Big(\langle \boldsymbol{Z}_i, \boldsymbol{B} - \boldsymbol{X}_i\rangle + B_\phi(\boldsymbol{B}, \boldsymbol{X}_i)\Big). \end{aligned} \tag{9}$$

where $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ are dual variables, B_ϕ denotes the Bregman divergence term determined by a convex function ϕ. When $\phi = \|\cdot\|_2$, $B_\phi(\boldsymbol{B}, \boldsymbol{X}_i) = \frac{1}{2}\|\boldsymbol{B} - \boldsymbol{X}_i\|_F^2$. When ϕ is the entropy function, $B_\phi(\boldsymbol{B}, \boldsymbol{X}_i) = \mathrm{KL}(\boldsymbol{B}\|\boldsymbol{X}_i)$.

After initializing $\boldsymbol{R}^{(0)} = \boldsymbol{X}_1^{(0)} = \boldsymbol{X}_2^{(0)} = \boldsymbol{B}_0$ and $\boldsymbol{Z}_1^{(0)} = \boldsymbol{Z}_2^{(0)} = \boldsymbol{0}_{N\times N}$, we update the variables iteratively by alternating optimization. In the t-th iteration, we have

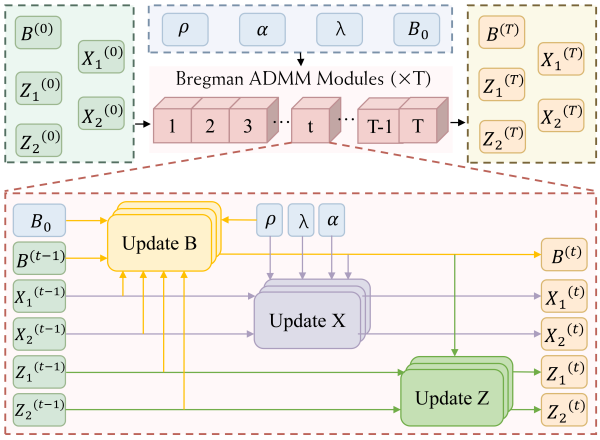


Figure 2: An illustration of the feed-forward step of our Bregman ADMM module.

- **Update $\boldsymbol{B}$** by solving the following problem:

$$\min_{\boldsymbol{B}\in\Omega} \mathrm{KL}(\boldsymbol{B}\|\boldsymbol{B}_0) + \rho \sum_{i=1,2} (\langle \boldsymbol{Z}_i^{(t)}, \boldsymbol{B}\rangle + \mathrm{KL}(\boldsymbol{B}\|\boldsymbol{X}_i^{(t)})),$$

where the Bregman divergence is set to be KL-divergence. This problem has a closed form solution:

$$\boldsymbol{B}^{(t+1)} = \sigma_r\Big(\frac{\log \boldsymbol{B}_0 + \rho\sum_{i=1,2}(\log \boldsymbol{X}_i^{(t)} - \boldsymbol{Z}_i^{(t)})}{1+2\rho}\Big).$$

- **Update $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$** via soft-thresholding: Setting the Bregman divergence to be the squared ℓ_2-norm, we have

$$\begin{aligned} &\min_{\boldsymbol{X}_1} \frac{\rho}{2}\|\boldsymbol{X}_1 - \boldsymbol{B}^{(t+1)}\|_2^2 - \rho\langle \boldsymbol{Z}_1^{(t)}, \boldsymbol{X}_1\rangle + \lambda\alpha\|\boldsymbol{X}_1\|_1 \\ &\Rightarrow \ \boldsymbol{X}_1^{(t+1)} = S_{\frac{\lambda\alpha}{\rho}}(\boldsymbol{B}^{(t+1)} + \boldsymbol{Z}_1^{(t)}), \end{aligned}$$

where $S_\tau(a) = \mathrm{sign}(a)(|a| - \tau)_+$ is the elementwise soft-thresholding function.

$$\begin{aligned} &\min_{\boldsymbol{X}_2} \frac{\rho}{2}\|\boldsymbol{X}_2 - \boldsymbol{B}^{(t+1)}\|_2^2 - \rho\langle \boldsymbol{Z}_2^{(t)}, \boldsymbol{X}_2\rangle + \lambda\alpha R(\boldsymbol{X}_2) \\ &\Rightarrow \begin{cases} R(\cdot) = \|\cdot\|_{1,2}: \ \boldsymbol{x}_{n,2}^{(t+1)} = \tau_n S_{\frac{\lambda\alpha}{\rho}}(\boldsymbol{b}_n^{(t+1)} + \boldsymbol{z}_{n,2}^{(t)}) \\ R(\cdot) = \|\cdot\|_*: \ \boldsymbol{X}_2^{(t+1)} = \boldsymbol{U} S_{\frac{\lambda(1-\alpha)}{\rho}}(\boldsymbol{\Sigma})\boldsymbol{V}^\top, \end{cases} \end{aligned}$$

where we denote $\boldsymbol{x}_{n,2}^{(t+1)}$, $\boldsymbol{z}_{n,2}^{(t)}$, and $\boldsymbol{b}_n^{(t+1)}$ as the n-th column of $\boldsymbol{X}_2^{(t+1)}$, $\boldsymbol{Z}_2^{(t)}$, and $\boldsymbol{B}^{(t+1)}$, respectively. $\tau_n = (1 - \frac{(1-\alpha)\lambda}{\rho\|S_{\frac{\lambda\alpha}{\rho}}(\boldsymbol{b}_n^{(t+1)} + \boldsymbol{z}_{n,2}^{(t)})\|_2})_+$. $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the singular value decomposition (SVD) of $\boldsymbol{B}^{(t+1)} + \boldsymbol{Z}_2^{(t)}$. The closed-form solutions are derived based on the implementations of $R(\cdot)$.

- **Update $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$** by the following step:

$$\begin{aligned} \boldsymbol{Z}_1^{(t+1)} &= \boldsymbol{Z}_1^{(t)} + (\boldsymbol{B}^{(t+1)} - \boldsymbol{X}_1^{(t+1)}), \\ \boldsymbol{Z}_2^{(t+1)} &= \boldsymbol{Z}_2^{(t)} + (\boldsymbol{B}^{(t+1)} - \boldsymbol{X}_2^{(t+1)}). \end{aligned} \tag{10}$$

Repeating the above updates T times till the objective function converges, we obtain the final event branch matrix.

For Hawkes process, we can apply the above algorithm directly in the E-step and obtain a structured responsibility matrix accordingly. For neural TPPs, we follow the work in (Xu and Cheng 2023), unrolling the iterations as T layers. Accordingly, the proposed Bregman ADMM module can be implemented as the stack of the T layers, as illustrated in Figure 2. Plugging the Bregman ADMM module into the attention layer leads to a new attention mechanism. Note that, the T layers are involved in the back-propagation when learning the neural TPPs. When $R(\cdot) = \|\cdot\|_*$, we detach the gradient of $\boldsymbol{X}_2$ such that the SVD operation will not be considered and the memory cost can be saved.

Comparisons with Existing Methods

Our BADMM module is applicable to improve both traditional EM paradigms and Transformer-based TPP models for inferring event branches. To our knowledge, another plug-and-play module driven by optimization algorithm and with a similar functionality is the Sinkhorn-based module proposed in (Mena et al. 2020; Sander et al. 2022). Specifically, focusing on the EM algorithm of Gaussian mixture models, the work in (Mena et al. 2020) revisits the E-step of the algorithm and computes the responsibility matrices by the Sinkhorn-scaling algorithm (Sinkhorn and Knopp 1967). Similarly, the Sinkformer in (Sander et al. 2022) improves the attention layer of Transformer encoder, applying the Sinkhorn-scaling algorithm to derive attention maps. These methods impose a doubly-stochastic constraint on the responsibility matrices and attention maps, treating the matrices as entropic optimal transport plans (Cuturi 2013).

Note that although the Sinkhorn-based module can also derive structured matrices, it is inapplicable for inferring event branches of TPPs. For the EM algorithm of Hawkes processes, the responsibility matrix is a lower-triangular matrix. Applying the Sinkhorn-scaling algorithm to the matrix leads to a diagonal matrix. It means that all the events are independent, which is unreasonable in practice. For the attention maps in neural TPPs, although we can apply the Sinkhorn-based attention layer first and then mask the doubly-stochastic attention map to a lower-triangular matrix, this operation breaks the row-wise normalization structure of the attention map. As a result, we can no longer interpret the attention map as the posterior probabilities of historical events given the current ones.

Experiments

We conducted comprehensive experiments on both synthetic and real-world datasets and evaluate the usefulness of our module. In addition, we analyzed the interpretabilty of inferred event branches and demonstrated the robustness of our module to its hyperparameters. All the experiments are run on two 3090 GPUs, and we record each method's averaged performance and standard deviation in three trials.

Implementation Details

Datasets We apply one synthetic event sequence dataset and four real-world ones in our experiments, whose statistics are shown in Table 1. The synthetic dataset **Conttime** (Mei and Eisner 2017) is simulated by Hawkes process, while the real-world datasets, including **Retweet** (Zhao et al. 2015), **StackOverflow (SO)** (Jure 2014), **Amazon** (Xue et al. 2023), and **Taobao** (Xue et al. 2023), contain user behaviors on different platforms, which are commonly used for evaluating TPP models. In addition, to evaluate the rationality of inferred event branches, we consider the dialogue data in the movie "**12 Angry Men**" (Zhang et al. 2018), in which each juror is treated as an event type and the timestamped sentences of the jurors are events.

Dataset	# Event Types	# Sequences	Max. length
Conttime	5	10,000	100
Taobao	17	2,000	94
Retweet	3	24,000	264
StackOverflow	22	6,633	736
Amazon	16	12,556	282
12 Angry Men	12	1	587

Table 1: The statistics of datasets

Baselines We plug our BADMM module into existing TPPs and evaluate its impacts accordingly. Here, we consider the classic Hawkes process (**HP** in (Zhou, Zha, and Song 2013b)) and two Transformer-based TPPs (**THP** in (Zuo et al. 2020) and **SAHP** in (Zhang et al. 2020)). For HP, we apply our BADMM module to estimate its responsibility matrix in the EM algorithm. For THP and SAHP, we replace their self-attention layer with our BADMM module. The baselines of our method include: i) the HP learned by EM algorithm, ii) the HP-based source tracking (**HPST**) in (Zhang et al. 2018), iii) the THP and SAHP learned by SGD, and iv) the THP and SAHP applying the Sinkhorn-based attention layers in (Sander et al. 2022).

Hyperparameter settings For our BADMM module, we consider two implementations, denoted as $\textbf{BADMM}_*$ (i.e., $R(\cdot) = \|\cdot\|_*$) and $\textbf{BADMM}_{1,2}$ (i.e., $R(\cdot) = \|\cdot\|_{1,2}$), respectively. The weight ρ of Lagrangian term only affects the convergence rate, so we simply set it to be 1 in our experiments. For the weight of regularizer, we set $\alpha \in (0, 1)$ and $\lambda \in \{0.01, 0.1, 1, 10, 100\}$. Applying grid search, we find their optimal configurations. For the number of iterations T, we follow the setting of Sinkhorn-based module in (Sander et al. 2022) and set $T = 2$, which achieves a trade-off between the model complexity and capability.

Evaluation metrics When evaluating each learned model in a testing set, we record i) the log-likelihood per event (**ELL**) and ii) the prediction accuracy of event types (**ACC**). In addition, to evaluate the rationality of inferred event branches, we visualize the transition matrices learned by different methods and check the learned important events and their triggering patterns qualitatively.

Effectiveness and Rationality

Impacts on Model Performance We conducted a thorough evaluation of BADMM-enhanced methods against various baselines across multiple datasets. The results in Table 2

Model	Method	Taobao		Retweet		StackOverflow		Amazon		Conttime	
		ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑
HP	Classic EM	$-0.262_{0.014}$	$0.463_{0.001}$	$-9.022_{0.017}$	$0.459_{0.008}$	$-3.042_{0.089}$	$0.439_{0.002}$	$-2.561_{0.000}$	$0.355_{0.002}$	$-1.076_{0.001}$	$0.340_{0.001}$
	$\text{BADMM}_{1,2}$	$0.220_{0.000}$	$0.476_{0.001}$	$-8.902_{0.001}$	$0.564_{0.000}$	$-2.758_{0.012}$	$0.452_{0.002}$	$-2.561_{0.000}$	$0.366_{0.001}$	$-1.073_{0.000}$	$0.340_{0.001}$
	BADMM_{*}	$\mathbf{0.223}_{0.000}$	$\mathbf{0.480}_{0.001}$	$\mathbf{-8.902}_{0.000}$	$\mathbf{0.565}_{0.000}$	$\mathbf{-2.744}_{0.010}$	$\mathbf{0.454}_{0.002}$	$\mathbf{-2.561}_{0.000}$	$\mathbf{0.368}_{0.000}$	$\mathbf{-1.073}_{0.000}$	$\mathbf{0.340}_{0.001}$
SAHP	Softmax	$-9.879_{0.856}$	$0.434_{0.000}$	$-7.878_{0.045}$	$0.537_{0.001}$	$-3.614_{0.247}$	$0.353_{0.012}$	$-3.223_{0.031}$	$0.295_{0.052}$	$-1.416_{0.104}$	$0.264_{0.037}$
	Sinkhorn	$-9.767_{0.552}$	$0.434_{0.000}$	$-7.702_{0.488}$	$0.541_{0.005}$	$-3.448_{0.097}$	$0.356_{0.010}$	$-3.137_{0.283}$	$0.306_{0.029}$	$-1.363_{0.181}$	$0.272_{0.026}$
	$\text{BADMM}_{1,2}$	$-9.616_{0.485}$	$0.434_{0.000}$	$-7.802_{0.462}$	$0.538_{0.001}$	$-3.625_{0.448}$	$0.356_{0.012}$	$-3.133_{0.271}$	$0.299_{0.036}$	$-1.305_{0.102}$	$\mathbf{0.283}_{0.029}$
	BADMM_{*}	$\mathbf{-9.594}_{0.119}$	$\mathbf{0.434}_{0.000}$	$\mathbf{-7.686}_{0.090}$	$\mathbf{0.542}_{0.003}$	$\mathbf{-3.374}_{0.062}$	$\mathbf{0.365}_{0.003}$	$\mathbf{-3.127}_{0.282}$	$\mathbf{0.307}_{0.009}$	$\mathbf{-1.235}_{0.086}$	$0.278_{0.022}$
THP	Softmax	$0.170_{0.006}$	$0.436_{0.000}$	$-9.373_{0.716}$	$0.511_{0.013}$	$-0.698_{0.007}$	$0.450_{0.002}$	$0.542_{0.001}$	$0.351_{0.001}$	$-0.390_{0.043}$	$0.696_{0.051}$
	Sinkhorn	$0.178_{0.007}$	$0.436_{0.000}$	$-11.020_{0.134}$	$0.525_{0.008}$	$-0.700_{0.010}$	$0.433_{0.004}$	$0.288_{0.025}$	$0.331_{0.022}$	$-0.394_{0.039}$	$0.633_{0.036}$
	$\text{BADMM}_{1,2}$	$0.190_{0.013}$	$0.436_{0.000}$	$-8.961_{0.233}$	$0.535_{0.003}$	$-0.696_{0.008}$	$0.453_{0.003}$	$0.542_{0.001}$	$0.355_{0.001}$	$-0.390_{0.035}$	$0.727_{0.031}$
	BADMM_{*}	$\mathbf{0.193}_{0.012}$	$\mathbf{0.436}_{0.000}$	$\mathbf{-8.753}_{0.046}$	$\mathbf{0.538}_{0.001}$	$\mathbf{-0.696}_{0.002}$	$\mathbf{0.453}_{0.002}$	$\mathbf{0.543}_{0.000}$	$\mathbf{0.356}_{0.001}$	$\mathbf{-0.385}_{0.041}$	$\mathbf{0.748}_{0.025}$

Table 2: Comparisons for various methods on model performance.

Rank	GPT-4o	HPST	BADMM		
			HP	THP	SAHP
1	Juror 8	Juror 8	Juror 8	Juror 8	Juror 8
2	Juror 3	Juror 3	Juror 3	Juror 3	Juror 3
3	Juror 7	Juror 7	Juror 7	Juror 7	Juror 7
4	Juror 10	Juror 1	Juror 10	Juror 10	Juror 4
5	Juror 4	Juror 10	Juror 12	Juror 1	Juror 10

Table 3: Top-5 most influential jurors in "12 Angry Men"

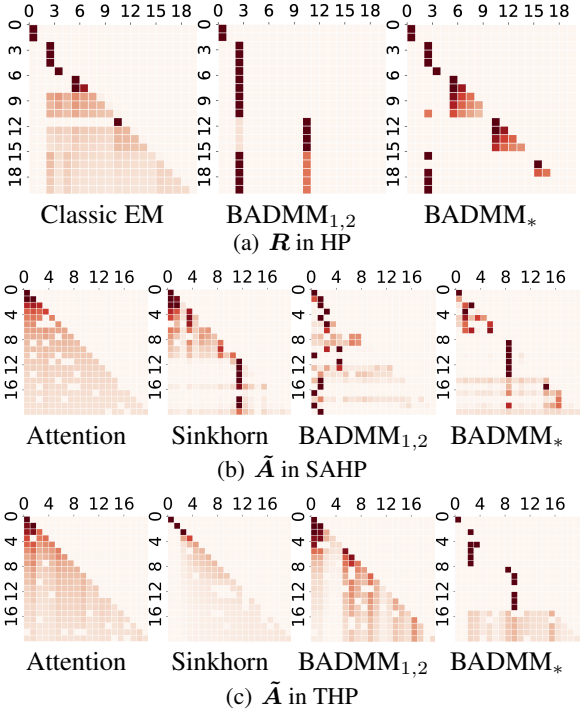
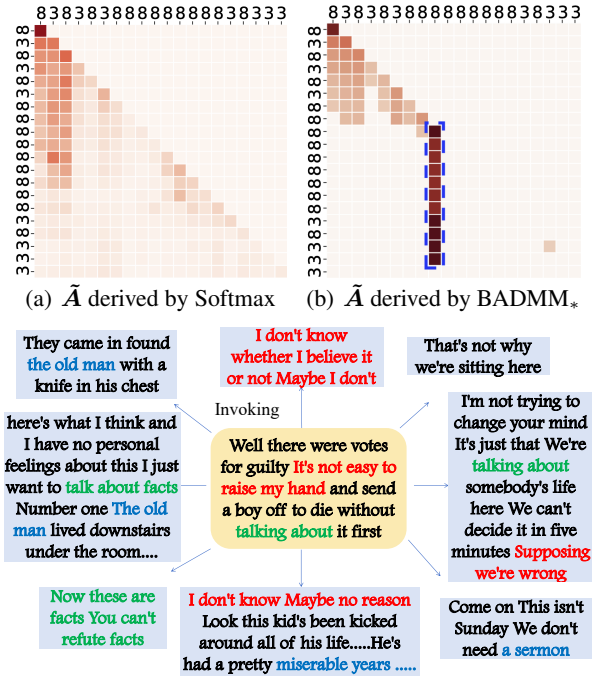


Figure 3: Visualization of inferred transition matrices. The event sequence is from the Conttime dataset.

demonstrate the consistent effectiveness of our BADMM module across all datasets and backbone models. In most situations, applying our BADMM module significantly improves the performance of both classical and neural TPPs. In our opinion, this phenomenon is because the event sequences in practice are driven by hidden branching processes, so that inferring event branches during training does not do harm to the learning of TPPs. Moreover, the existing TPPs, especially those neural ones, may suffer from the over-fitting issue during training. Our BADMM module imposes structural constraints on their transition matrices, regularizing model parameters implicitly to mitigate the issue.

Qualitative comparisons In Figure 3, we visualize the transition matrices inferred by different methods. We can find that the original transition matrices are over-smoothed, which contain too many nonzero elements. In contrast, applying our BADMM module indeed leads to sparse and low-rank transition matrices for both classic and neural TPPs, which enhances the structures of transition matrices significantly. For neural TPPs, we further compare the transition matrices derived by our BADMM module with those derived by the Sinkhorn module (Sander et al. 2022). The Sinkhorn module first derives a doubly stochastic matrix and then masks it to a lower triangular matrix. Although it can achieve low-rank and sparse transition matrices to some extent, it breaks the row-normalization constraint of the transition matrix and thus results in inferior performance compared to our method (as shown in Table 2).

In addition, the two implementations of BADMM module often lead to different event branches. Because of using sparse group-lasso regularization, $\text{BADMM}_{1,2}$ tends to very sparse transition matrices in which only some columns have dense nonzero elements. It means that $\text{BADMM}_{1,2}$ attributes the generation of the event sequence from few globally-significant events. On the other side, BADMM_{*} preserves the structures of the initial transition matrices better and tends to identify the events having local impacts on its subsequent events. According to the results in Table 2, we can find that BADMM_{*} often outperforms $\text{BADMM}_{1,2}$ on data fitness and prediction accuracy, which means that the transition matrices of BADMM_{*} are often associated with



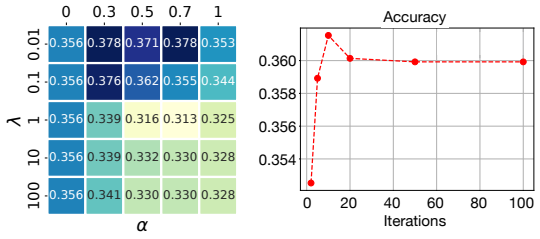
(a) $\tilde{\boldsymbol{A}}$ derived by Softmax (b) $\tilde{\boldsymbol{A}}$ derived by BADMM$_*$

(c) Triggering patterns among key dialogues

Figure 4: (a, b) Inferred event branches. (c) The branch triggered by the key sentence (corresponding to the blue region in (b)).

the models with low risks of misspecification. Considering this fact, we think BADMM$_*$ can generate more reasonable event branches and reveal triggering patterns among events better. In the following experiments, we mainly test the performance of BADMM$_*$.

Rationality of Inferred Event Branches For the dialogue in the movie "12 Angry Men", we follow the HPST method (Zhang et al. 2018), learning a transformer hawkes process and inferring a transition matrix for the jurors' sentences. According to the learned transformer hawkes process, we can rank the 12 jurors in the movie based on the overall impacts of their sentences (i.e., $\mathbf{1}^\top \boldsymbol{B}\boldsymbol{S}$, where $\mathbf{1}^\top \boldsymbol{B} \in \mathbb{R}^N$ records the impacts of N sentences and $\boldsymbol{S} \in \{0,1\}^{N\times 12}$ maps the sentences to corresponding jurors). Ideally, we hope that the ranking result can reflect the significance of the jurors in the movie. Taking the ranking result derived by GPT-4o as the ground truth (which is checked by five volunteers watched the movie before), we compare the ranking results derived by the TPPs learned with our BADMM module with that achieved by HPST. Table 3 lists the top-5 most influential jurors proposed by different methods. Both our method and HPST can detect the top-2 most influential jurors (i.e., Jurors 8 and 3, who insist on acquittal and conviction, respectively). Compared to HPST, the ranking results of our method are more consistent with that of GPT-4o. For example, the SAHP learned with our BADMM module considers Juror 4 in the top-5 list. This juror appears in the list of GPT-4o but is missed by HPST. Note that both



(a) ACC of HP (b) Iterations v.s. ACC

Figure 5: Robustness test of our BADMM module.

GPT-4o and HPST leverage textual information when ranking the jurors, while our method purely relies on the timestamps and the jurors' IDs within the event sequence. In other words, by applying our BADMM module, we can learn reliable event branches that are consistent with those estimated by the text-based models.

To further verify the rationality of inferred event branches, we sample the rows and columns of the transition matrix relevant to Jurors 8 and 3 and visualize it in Figure 4. Similar to the results in Figure 3, we can find that compared to the Softmax operation of THP, our BADMM module can derive a transition matrix with better sparsity, whose event branches are more clear. In addition, based on the inferred transition matrix, we show the triggering patterns hidden the dialogue between the two jurors. We can find that the triggering patterns are consistent with the semantics of the sentences.

Robustness to Hyperparameters

The impacts of key hyperparameters (α, λ, and the number of iterations T) on model performance are shown in Figure 5. Specifically, Figure 5(a) shows the ACC of the models learned with different configurations of α and λ. The results show that our BADMM module consistently performs well within a broad range of the hyperparameter settings, indicating that our method is robust to moderate changes in these hyperparameters. Figure 5(b) shows the effect of the number of iterations on the performance of our method. We can find that our BADMM module quickly converges within a few iterations, with marginal gains observed beyond a certain threshold. This rapid convergence implies that, in practical scenarios, we can implement our BADMM module using limited iterations, which reduces computational costs while maintaining high performance.

Conclusion

We introduced a novel BADMM module, which helps infer structured event branches for various TPPs. By integrating this module into both classic and neural TPPs, we addressed the common over-smoothness issue of event branch inference, providing an effective and robust solution to impose sparse and low-rank structures on transition matrices. Our method improves the interpretability of learned TPPs and enhances their model performance.

Acknowledgments

This work was supported by National Natural Science Foundation (62106271, 92270110), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China. We also acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China and by the funds from Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, and from Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China.

References

Bacry, E.; Mastromatteo, I.; and Muzy, J.-F. 2015. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01): 1550005.

Champion, T.; Da Costa, L.; Bowman, H.; and Grześ, M. 2022. Branching time active inference: the theory and its generality. *Neural Networks*, 151: 295–316.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Dion, J.-P.; and Yanev, N. 1994. Statistical inference for branching processes with an increasing random number of ancestors. *Journal of statistical planning and inference*, 39(2): 329–351.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1555–1564.

Elhamifar, E.; and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11): 2765–2781.

Farajtabar, M.; Du, N.; Gomez-Rodriguez, M.; Valera, I.; Zha, H.; and Song, L. 2014. Shaping social activity by incentivizing users. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2474–2482.

Farajtabar, M.; Rodriguez, M. G.; Zamani, M.; Du, N.; Zha, H.; and Song, L. 2015. Back to the past: Source identification in diffusion networks from partially observed cascades. In *Artificial Intelligence and Statistics*, 232–240. PMLR.

Farajtabar, M.; Wang, Y.; Gomez-Rodriguez, M.; Li, S.; Zha, H.; and Song, L. 2017a. Coevolve: A joint point process model for information diffusion and network evolution. *Journal of Machine Learning Research*, 18(41): 1–49.

Farajtabar, M.; Yang, J.; Ye, X.; Xu, H.; Trivedi, R.; Khalil, E.; Li, S.; Song, L.; and Zha, H. 2017b. Fake news mitigation via point process based intervention. In *International conference on machine learning*, 1097–1106. PMLR.

González, M.; Gutiérrez, C.; Martínez, R.; and Del Puerto, I. M. 2013. Bayesian inference for controlled branching processes through MCMC and ABC methodologies. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas*, 107(2): 459–473.

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

Isham, V.; and Westcott, M. 1979. A self-correcting point process. *Stochastic processes and their applications*, 8(3): 335–347.

Jure, L. 2014. SNAP Datasets: Stanford large network dataset collection. *Retrieved December 2021 from http://snap. stanford. edu/data*.

Kingman, J. F. C. 1992. *Poisson processes*, volume 3. Clarendon Press.

Lewis, P. W.; and Shedler, G. S. 1979. Simulation of nonhomogeneous Poisson processes by thinning. *Naval research logistics quarterly*, 26(3): 403–413.

Liu, G.; Lin, Z.; and Yu, Y. 2010. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 663–670.

Mei, H.; and Eisner, J. 2017. The neural hawkes process: a neurally self-modulating multivariate point process. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6757–6767.

Mena, G.; Nejatbakhsh, A.; Varol, E.; and Niles-Weed, J. 2020. Sinkhorn em: an expectation-maximization algorithm based on entropic optimal transport. *arXiv preprint arXiv:2006.16548*.

Møller, J.; and Rasmussen, J. G. 2006. Approximate simulation of Hawkes processes. *Methodology and Computing in Applied Probability*, 8: 53–64.

Ross, S. M.; Kelly, J. J.; Sullivan, R. J.; Perry, W. J.; Mercer, D.; Davis, R. M.; Washburn, T. D.; Sager, E. V.; Boyce, J. B.; and Bristow, V. L. 1996. *Stochastic processes*, volume 2. Wiley New York.

Sander, M. E.; Ablin, P.; Blondel, M.; and Peyré, G. 2022. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, 3515–3530. PMLR.

Simon, N.; Friedman, J.; Hastie, T.; and Tibshirani, R. 2013. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2): 231–245.

Sinkhorn, R.; and Knopp, P. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2): 343–348.

Wang, H.; and Banerjee, A. 2014. Bregman alternating direction method of multipliers. *Advances in Neural Information Processing Systems*, 27.

Wang, Q.; Cheng, M.; Yuan, S.; and Xu, H. 2023. Hierarchical contrastive learning for temporal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10166–10174.

Xu, H.; Carin, L.; and Zha, H. 2018. Learning registered point processes from idiosyncratic observations. In *International Conference on Machine Learning*, 5443–5452. PMLR.

Xu, H.; and Cheng, M. 2023. Regularized optimal transport layers for generalized global pooling operations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu, H.; Farajtabar, M.; and Zha, H. 2016. Learning granger causality for hawkes processes. In *International Conference on Machine Learning*, 1717–1726. PMLR.

Xu, H.; Wu, W.; Nemati, S.; and Zha, H. 2016. Patient flow prediction via discriminative learning of mutually-correcting processes. *IEEE transactions on Knowledge and Data Engineering*, 29(1): 157–171.

Xue, S.; Shi, X.; Chu, Z.; Wang, Y.; Zhou, F.; Hao, H.; Jiang, C.; Pan, C.; Xu, Y.; Zhang, J. Y.; et al. 2023. Easytpp: Towards open benchmarking the temporal point processes. *arXiv preprint arXiv:2307.08097*.

Yang, Y.; Sun, J.; Li, H.; and Xu, Z. 2018. ADMM-CSNet: A deep learning approach for image compressive sensing. *IEEE transactions on pattern analysis and machine intelligence*, 42(3): 521–538.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes process. In *International conference on machine learning*, 11183–11193. PMLR.

Zhang, W.; Bu, F.; Owens-Oas, D.; Heller, K.; and Zhu, X. 2018. Who started it? identifying root sources in textual conversation threads. *arXiv preprint arXiv:1809.03648*.

Zhao, Q.; Erdogdu, M. A.; He, H. Y.; Rajaraman, A.; and Leskovec, J. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1513–1522.

Zhou, K.; Zha, H.; and Song, L. 2013a. Learning triggering kernels for multi-dimensional hawkes processes. In *International conference on machine learning*, 1301–1309. PMLR.

Zhou, K. e.; Zha, H.; and Song, L. 2013b. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, 641–649. PMLR.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer hawkes process. In *International conference on machine learning*, 11692–11702. PMLR.