Learning Mixture of Neural Temporal Point Processes for Multi-dimensional Event Sequence Clustering

Yunhao Zhang[1] , **Junchi Yan**[1*] , **Xiaolu Zhang**[2] , **Jun Zhou**[2] and **Xiaokang Yang**[1]
[1]Department of CSE, MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
[2]Ant Financial Services Group
{zhangyunhao, yanjunchi, xkyang}@sjtu.edu.cn, {yueyin.zxl, jun.zhoujun}@antfin.com

Abstract

Multi-dimensional event sequence clustering applies to many scenarios e.g. e-Commerce and electronic health. Traditional clustering models fail to characterize complex real-world processes due to the strong parametric assumption. While Neural Temporal Point Processes (NTPPs) mainly focus on modeling similar sequences instead of clustering. To fill the gap, we propose Mixture of Neural Temporal Point Processes (NTPP-MIX), a general framework that can utilize many existing NTPPs for multi-dimensional event sequence clustering. In NTPP-MIX, the prior distribution of coefficients for cluster assignment is modeled by a Dirichlet distribution. When the assignment is given, the conditional probability of a sequence is modeled by the mixture of a series of NTPPs. We combine variational EM algorithm and Stochastic Gradient Descent (SGD) to efficiently train the framework. In E-step, we fix parameters for NTPPs and approximate the true posterior with variational distributions. In M-step, we fix variational distributions and use SGD to update parameters of NTPPs. Extensive experimental results on four synthetic datasets and three real-world datasets show the effectiveness of NTPP-MIX against state-of-the-arts.

1 Introduction

Many real-world processes can be characterized by time-stamped multi-typed[1] event sequences, e.g. e-Commerce [Xu *et al.*, 2014] and electronic health records [Enguehard *et al.*, 2020]. Given a set of event sequences, a natural task is to cluster them into groups such that sequences generated by similar underlying dynamics are grouped. Multi-dimensional event sequence clustering is important to downstream applications. For instance, clustering user's behaviors in e-Commerce helps to promote recommendation system.

Despite the importance, research on event sequence clustering (especially for multi-dimensional events i.e. typed events) is still limited. [Xu and Zha, 2017] propose a Dirichlet Mixture model of Hawkes Processes (DMHP) where event sequences are characterized via Hawkes processes [Hawkes, 1971] with different parameters. However, as a traditional Temporal Point Process (TPP), it is hard for Hawkes process to model complex real-world dynamics due to its limited parametric assumption. The recent work [Wu *et al.*, 2022] takes a deep reinforcement learning view of event sequence clustering. But it only works for uni-dimensional event sequences and its generative adversarial learning framework can be inefficient and less stable.

With the development of deep learning, many Neural Temporal Point Processes (NTPPs) have been developed. Recurrent Neural Networks (RNN) [Du *et al.*, 2016; Mei and Eisner, 2016], Neural Ordinary Differential Equations [Jia and Benson, 2019; Chen *et al.*, 2021] and Transformers [Zhang *et al.*, 2020; Zuo *et al.*, 2020] are used to model the conditional intensity function of event sequences. [Shchur *et al.*, 2020; Sharma *et al.*, 2021; Omi *et al.*, 2019] propose intensity-free NTPPs to directly model the conditional distribution of inter-event times or cumulative hazard function. Many Maximum Likelihood Estimation free (MLE-free) NTPPs are also developed [Xiao *et al.*, 2017; Upadhyay *et al.*, 2018; Yan *et al.*, 2018; Li *et al.*, 2018]. Although these NTPPs are more flexible than Hawkes process, they can not perform clustering because they assume all observed sequences are generated by similar dynamics to which they strike to fit.

We propose mixture of neural temporal point processes (NTPP-MIX), a general framework that can incorporate various MLE-based NTPPs for multi-dimensional event sequence clustering. Specifically, the prior of the mixing coefficients is modeled by a Dirichlet distribution. Then the coefficients are used to model the prior of cluster assignment. For each sequence, its conditional probability on given assignment is modeled via a series of NTPPs. Our goal is to maximize the marginal probability and get the posterior of cluster assignment. Therefore we introduce variational distribution to approximate the posterior and combine variational EM and Stochastic Gradient Descent (SGD) to efficiently train the framework. In E-step, we fix parameters for NTPPs and use variational distributions to approximate the true posterior. While in M-step, we fix variational distributions and update

[*]Correspondence author: Junchi Yan. The work was in part supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), National NSFC (U19B2035, 61972250, 72061127003), and funding from Ant Group.

[1]We use the term 'multi-dimensional' & 'typed' interchangeably.

parameters for NTPPs using SGD. **The contributions are:**

1) We develop a general framework NTPP-MIX for multi-dimensional event sequence clustering using neural networks. To our best knowledge, it is the first neural-network-based framework for multi-dimensional event sequence clustering and can utilize various previous MLE-based NTPPs as its inside module. To put it in the context, the work [Xu and Zha, 2017] is limited to the parametric model, neural model [Wu *et al.*, 2022] can only handle uni-dimensional events.

2) We propose a training algorithm based on variational EM and SGD to efficiently train the framework and give a theoretical proof of its correctness and convergence.

3) Extensive experimental results on both synthetic and real-world benchmarks show the effectiveness of our NTPP-MIX framework. Source code will be made public available.

2 Preliminaries and Related Works

Traditional temporal point processes Temporal Point Process (TPP) [Daley and David, 2007] is a useful tool to model a set of similar event sequences. We denote a multi-dimensional event sequence as $S = \{(t_i, v_i)\}_{i=1}^{L}$, where $t_i \in [0, T)$ is timestamp and $v_i \in \{1, 2, \ldots, C\}$ is event type. A TPP first defines the conditional intensity function:

$$\lambda(t, u|\mathcal{H}_t) = \lim_{dt \to 0} \frac{\mathbb{P}\left(N_u(t + dt) - N_u(t) = 1|\mathcal{H}_t\right)}{dt} \quad (1)$$

where $N_u(t)$ denotes the counting process which counts the number of events of type u until time t, $\mathcal{H}_t = \{(t_i, v_i) : t_i < t\}$ represents the history up to t. The log likelihood of S is:

$$\log p(S) = \sum_{i=1}^{L} \log \lambda(t_i, v_i|\mathcal{H}_{t_i}) - \sum_{u=1}^{C} \int_0^{t_L} \lambda(t, u|\mathcal{H}_t)dt \quad (2)$$

Most traditional TPP models design a parametric form for intensity, which is often trained via MLE [Yan *et al.*, 2016].

Neural temporal point processes Neural Temporal Point Processes (NTPPs) have been recently proposed to capture complex dynamics. [Du *et al.*, 2016] use RNNs to give a flexible form of intensity. [Mei and Eisner, 2016] further improve NTPP by constructing continuous-time RNNs. [Jia and Benson, 2019; Chen *et al.*, 2018] use the continuous-depth Neural Ordinary Differential Equations [Chen *et al.*, 2018] to achieve more flexible continuous-time state evolution. Transformer or Attention mechanism is used in [Zhang *et al.*, 2020; Zuo *et al.*, 2020] to capture long-term dependency.

The above NTPPs model the conditional intensity, while there are intensity-free NTPPs. [Omi *et al.*, 2019] model the integral of the intensity. [Shchur *et al.*, 2020; Sharma *et al.*, 2021] directly model the distribution of inter-event times.

Both intensity-based and intensity-free NTPPs are trained via MLE. Besides MLE, there are also MLE-free [Xiao *et al.*, 2017; Yan *et al.*, 2018; Li *et al.*, 2018] NTPPs.

Although these NTPPs are more flexible than traditional TPPs, they can not directly perform clustering as they assume observed sequences are generated by similar dynamics.

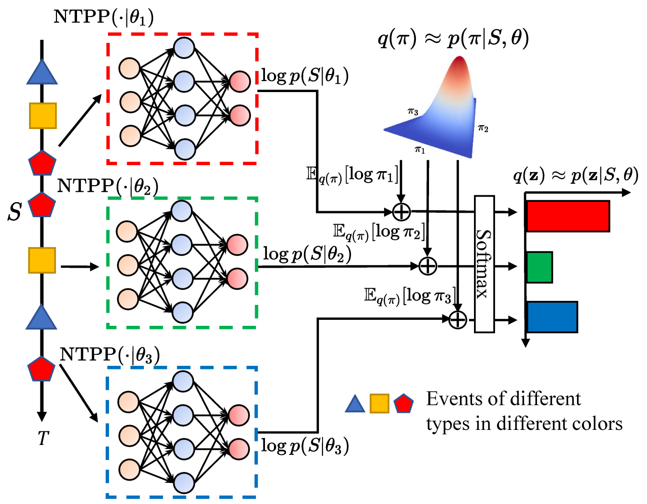

NTPP$(\cdot|\theta_1)$
NTPP$(\cdot|\theta_2)$
NTPP$(\cdot|\theta_3)$
S
T
$\log p(S|\theta_1)$
$\log p(S|\theta_2)$
$\log p(S|\theta_3)$
$q(\pi) \approx p(\pi|S, \theta)$
$\mathbb{E}_{q(\pi)}[\log \pi_1]$
$\mathbb{E}_{q(\pi)}[\log \pi_2]$
$\mathbb{E}_{q(\pi)}[\log \pi_3]$
Softmax
$q(\mathbf{z}) \approx p(\mathbf{z}|S, \theta)$
Events of different types in different colors

Figure 1: The proposed NTPP-MIX framework for $K = 3$. After training, an event sequence is input into K NTPPs to get conditional probabilities. Adding corresponding expectation of mixing coefficients, we can get the approximated posterior of assignment. Index n is omitted. Details of the derivation is in Sec. 3.2.

Event sequence clustering methods. Although event sequence modeling is well-studied, there are few studies about event sequence clustering. [Wu *et al.*, 2022] propose a reinforcement learning model but it only works for uni-dimensional event sequences. To our best knowledge, the most related work to ours is DMHP [Xu and Zha, 2017] which mixes Hawkes processes for clustering. Clusters are defined as Hawkes processes with different parameters. However, real-world data may be complex and can not be well modeled by Hawkes processes, e.g. a new event can inhibit other events instead of the triggering assumption of Hawkes.

3 NTPP-MIX: Mixture of NTPPs for Multi-dimensional Event Clustering

In this section, we propose mixture of neural temporal point processes (NTPP-MIX) framework. It combines NTPP and variational inference for multi-dimensional event clustering.

3.1 Model Framework

Given multi-dimensional event sequences $\mathbf{S} = \{S_n\}_{n=1}^{N}$, where $S_n = \{(t_i, v_i)\}_{i=1}^{L_n}$ is a sequence defined in Sec. 2. Our goal is to divide these N sequences into K groups such that the sequences generated by similar dynamics are grouped. For each S_n, there is a corresponding one-hot vector $\mathbf{z}_n \in \{0, 1\}^K$ denoting the group it belongs to. Denote the hidden variables as $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ and our goal is to give a probabilistic estimation of $\mathbf{Z}$. We combine NTPPs and mixture model to enable a probabilistic framework for this clustering problem. Our NTPP-MIX is general and can utilize various NTPPs as its basic component as shown in Fig. 1.

We first assume the prior of mixing coefficients $\boldsymbol{\pi}$ follows:

$$p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \quad (3)$$

which is a Dirichlet distribution, where $\boldsymbol{\alpha}_0 = [\alpha_0, \ldots, \alpha_0] \in \mathbb{R}^K$. Setting a prior for $\boldsymbol{\pi}$ results in a variational model that helps to select K [Bishop and Nasrabadi, 2007].

Given $\boldsymbol{\pi}$, the conditional distribution of $\mathbf{Z}$ is formed as: $p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}}$ where π_k is the k-th dimension of $\boldsymbol{\pi}$. This equals to $p(z_{nk} = 1|\boldsymbol{\pi}) = \pi_k$. Given $\mathbf{Z}$, we use K NTPPs to model the conditional probability of $\mathbf{S}$:

$$\begin{aligned} p(\mathbf{S}|\mathbf{Z}, \theta) &= \prod_{n=1}^{N} \prod_{k=1}^{K} p(S_n|\theta_k)^{z_{nk}} \\ p(S_n|\theta_k) &= \text{NTPP}(S_n|\theta_k) \end{aligned} \tag{4}$$

where $\theta = \{\theta_k\}_{k=1}^{K}$ and each θ_k denotes parameters of the k-th NTPP model. For an event sequence S_n, we input it into the k-th NTPP, and get the conditional probability $p(S_n|\theta_k)$. Note that NTPP here can be any MLE-based NTPP such as RMTPP [Du *et al.*, 2016], NHP [Mei and Eisner, 2016], SAHP [Zhang *et al.*, 2020], THP [Zuo *et al.*, 2020], LogNormMix [Shchur *et al.*, 2020], etc.

Summarizing above equations, we can get the joint distribution $p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)$. Our goal is to maximize marginal distribution $p(\mathbf{S}|\theta)$ w.r.t. θ and get the posterior $p(\mathbf{Z}|\mathbf{S}, \theta_{opt})$:

$$\begin{aligned} \theta_{opt} &= \arg\max_{\theta} p(\mathbf{S}|\theta) \\ &= \arg\max_{\theta} \iint p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta) d\mathbf{Z} d\boldsymbol{\pi} \\ p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta) &= p(\boldsymbol{\pi}) p(\mathbf{Z}|\boldsymbol{\pi}) p(\mathbf{S}|\mathbf{Z}, \theta) \\ &= \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \text{NTPP}(S_n|\theta_k)]^{z_{nk}} \end{aligned} \tag{5}$$

Note $\mathbf{Z}$ is discrete and we use integration for notation brevity.

3.2 Training via Variational EM and SGD

It is hard to optimize marginal probability $p(\mathbf{S}|\theta)$ and get the posterior $p(\mathbf{Z}|\mathbf{S}, \theta_{opt})$ directly as Eq. 5 contains complex integral terms and parameters for networks. To address this problem, we combine variational EM and SGD for training.

We introduce a variational distribution $q(\mathbf{Z}, \boldsymbol{\pi})$ to approximate $p(\mathbf{Z}, \boldsymbol{\pi}|\mathbf{S}, \theta)$. Using $q(\mathbf{Z}, \boldsymbol{\pi})$, we can get:

$$\mathcal{L}(q, \theta) = \log p(\mathbf{S}|\theta) - \text{KL}(q \| p) \tag{6}$$

where we define:

$$\begin{aligned} \mathcal{L}(q, \theta) &= \iint q(\mathbf{Z}, \boldsymbol{\pi}) \log \left[\frac{p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)}{q(\mathbf{Z}, \boldsymbol{\pi})} \right] d\mathbf{Z} d\boldsymbol{\pi} \\ \text{KL}(q \| p) &= \iint q(\mathbf{Z}, \boldsymbol{\pi}) \log \left[\frac{q(\mathbf{Z}, \boldsymbol{\pi})}{p(\mathbf{Z}, \boldsymbol{\pi}|\mathbf{S}, \theta)} \right] d\mathbf{Z} d\boldsymbol{\pi} \end{aligned} \tag{7}$$

Therefore, maximizing $\mathcal{L}(q, \theta)$ with respect to q and θ is equivalent to maximizing $\log p(\mathbf{S}|\theta)$ and minimizing $\text{KL}(q\|p)$. Maximizing $\log p(\mathbf{S}|\theta)$ makes our model characterize observed sequences well. Minimizing $\text{KL}(q\|p)$ approximates the posterior. Our goal becomes $\max_{\theta,q} \mathcal{L}(q, \theta)$.

In variational inference, $\mathcal{L}(q, \theta)$ is called evidence lower bound (ELBO) [Hoffman *et al.*, 2013]. Using mean field approximation [Opper and Saad, 2001], we assume $q(\mathbf{Z}, \boldsymbol{\pi}) = q(\mathbf{Z})q(\boldsymbol{\pi})$ and maximize $\mathcal{L}(q, \theta)$ with respect to $q(\mathbf{Z}), q(\boldsymbol{\pi}), \theta$ iteratively according to the following process.

Update $q(\mathbf{Z})$ (E-step) Fixing $q(\boldsymbol{\pi})$ and θ, viewing terms irrelevant to $q(\mathbf{Z})$ as constant, we can derive:

$$\begin{aligned} &\mathcal{L}(q, \theta) = \mathcal{L}(q(\mathbf{Z}), q(\boldsymbol{\pi}), \theta) \\ &= -\int q(\mathbf{Z}) \left(\log q(\mathbf{Z}) - \mathbb{E}_{q(\boldsymbol{\pi})}[\log p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)] \right) d\mathbf{Z} + C \\ &= -\text{KL}[q(\mathbf{Z}) \| \widetilde{p}(\mathbf{S}, \mathbf{Z}|\theta)] + C \end{aligned} \tag{8}$$

where we define a new distribution with $\log \widetilde{p}(\mathbf{S}, \mathbf{Z}|\theta) = \mathbb{E}_{q(\boldsymbol{\pi})}[\log p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)]$, the KL divergence is minimized by setting $q^*(\mathbf{Z}) = \widetilde{p}(\mathbf{S}, \mathbf{Z}|\theta)$:

$$\begin{aligned} &\log q^*(\mathbf{Z}) = \mathbb{E}_{q(\boldsymbol{\pi})}[\log p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)] \\ &= \mathbb{E}_{q(\boldsymbol{\pi})}[\log p(\mathbf{Z}|\boldsymbol{\pi})] + \log p(\mathbf{S}|\mathbf{Z}, \theta) + C \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \underbrace{\left\{ \mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] + \log p(S_n|\theta_k) \right\}}_{\log \rho_{nk}} + C \end{aligned} \tag{9}$$

Normalizing the above formulation, we obtain:

$$q^*(\mathbf{Z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}}, \quad r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{10}$$

Note that $q(\mathbf{Z})$ is an approximation of $p(\mathbf{Z}|\mathbf{S}, \theta)$, when the model is well-trained, $p(z_{nk} = 1|S_n, \theta_{opt}) \approx r_{nk}$, i.e. the posterior probability that S_n in group k is r_{nk}.

Update $q(\boldsymbol{\pi})$ (E-step). Fixing $q(\mathbf{Z})$ and θ, viewing terms irrelevant to $q(\boldsymbol{\pi})$ as constant, we can derive:

$$\mathcal{L}(q, \theta) = -\text{KL}[q(\boldsymbol{\pi}) \| \widetilde{p}(\mathbf{S}, \boldsymbol{\pi}|\theta)] + C \tag{11}$$

Similarly, $\log \widetilde{p}(\mathbf{S}, \boldsymbol{\pi}|\theta) = \mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)]$, the KL divergence is minimized by setting $q^*(\boldsymbol{\pi}) = \widetilde{p}(\mathbf{S}, \boldsymbol{\pi}|\theta)$:

$$\begin{aligned} &\log q^*(\boldsymbol{\pi}) = \mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{S}, \mathbf{Z}, \boldsymbol{\pi}|\theta)] \\ &= \log p(\boldsymbol{\pi}) + \mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{Z}|\pi)] + C \\ &= \sum_{k=1}^{K} \left(\alpha_0 + \sum_{n=1}^{N} r_{nk} - 1 \right) \log \pi_k + C \end{aligned} \tag{12}$$

Normalizing the above formulation, we recognize $q^*(\boldsymbol{\pi})$ is a Dirichlet distribution:

$$q^*(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}), \alpha_k = \alpha_0 + N_k, N_k = \sum_{n=1}^{N} r_{nk} \tag{13}$$

Update θ (M-step). Fixing $q(\mathbf{Z})$ and $q(\boldsymbol{\pi})$, viewing terms irrelevant to θ as constant, we can derive:

$$\begin{aligned} \mathcal{L}(q, \theta) &= \mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{S}|\mathbf{Z}, \theta)] + C \\ &= \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log p(S_n|\theta_k)}_{\mathcal{L}(\theta)} + C \end{aligned} \tag{14}$$

where $\mathcal{L}(\theta)$ denotes the term in $\mathcal{L}(q, \theta)$ that is only related to θ. As θ is the parameter for NTPPs, it is hard to obtain the optimal value of θ like $q(\mathbf{Z}), q(\boldsymbol{\pi})$. Instead, we use SGD with

$-\mathcal{L}(\theta)$ as objective function for optimization. Considering the problem is non-convex, we perform $M > 1$ steps of SGD:

$$\begin{aligned} &\theta^* = \theta^M, \theta^0 = \theta \\ &\theta^i = \theta^{i-1} + \gamma \nabla \mathcal{L}(\theta^{i-1}), i = 1, \ldots, M \end{aligned} \tag{15}$$

Although $\mathcal{L}(q, \theta)$ can not reach the maximum w.r.t. θ like the update of $q(\mathbf{Z}), q(\boldsymbol{\pi})$, it is still reasonable to assume that $\mathcal{L}(q, \theta)$ gets greater after M steps of SGD.

As each of the three steps w.r.t. $q(\mathbf{Z}), q(\boldsymbol{\pi}), \theta$ makes $\mathcal{L}(q, \theta)$ greater, performing them one after another, we get:

$$\mathcal{L}(q^{(t)}(\mathbf{Z}), q^{(t)}(\boldsymbol{\pi}), \theta^{(t)}) \geq \mathcal{L}(q^{(t-1)}(\mathbf{Z}), q^{(t-1)}(\boldsymbol{\pi}), \theta^{(t-1)})$$

where $q^{(t)}(\mathbf{Z}), q^{(t)}(\boldsymbol{\pi}), \theta^{(t)}$ denote distribution/parameters after iteration t. This suggests that after each iteration, $\mathcal{L}(q, \theta)$ gets greater than that of the last iteration. Performing these three steps iteratively, we can get the optimal solution.

Mini-batch training. Mini-batch training is often used to train NTPPs efficiently. We also incorporate it into our training algorithm. It is natural to update $q(\mathbf{Z})$ and θ with a batch of samples, but not for $q(\boldsymbol{\pi})$ as it is a global distribution for all samples. Inspired by the momentum technique [Sutskever *et al.*, 2013], we update α_k (parameter for $q(\boldsymbol{\pi})$) by:

$$\begin{aligned} \alpha_k^{(t)} &= (1-\eta)\alpha_k^{(t-1)} + \eta \alpha_k^{\mathbf{S}'} \\ \alpha_k^{\mathbf{S}'} &= \alpha_0 + \frac{N}{|\mathbf{S}'|} \sum_{S_n \in \mathbf{S}'} r_{nk} \end{aligned} \tag{16}$$

where $\mathbf{S}'$ is a batch of sequences sampled from the whole dataset. $\alpha_k^{\mathbf{S}'}$ is the parameter estimated on this batch. η is the update rate that conrtols how much to update with this batch.

Update the cluster number K. Thanks to the variational framework, NTPP-MIX has a natural tendency to set some mixture weights close to zero and choose a suitable number of effective components automatically. When K is unknown, we initialize K as a large number and remove redundant clusters during training. After each iteration of E-step and M-step, we can get $q^*(\boldsymbol{\pi})$ and its parameter $\boldsymbol{\alpha}$. As $q(\boldsymbol{\pi})$ is an approximation of $p(\boldsymbol{\pi}|\mathbf{S}, \theta)$, a small α_k indicates the effective number of sequences corresponding to cluster k is small, so we remove it. The removement also helps to speed up the training process as the number of NTPPs to fit is decreasing.

Detailed training algorithm is shown in Algorithm 1.

Algorithm 1: Training algorithm for NTPP-MIX

```
Input: Sequences $\mathbf{S} = \{S_n\}_{n=1}^N$. Initial number of
       clusters $K$ (to be automatically updated).
       Mini-batch size $B$. Training epochs $T$.
       Number of SGD steps to perform in each
       M-step $M$. $q(\boldsymbol{\pi})$ update rate $\eta$. Minimum
       cluster size $\epsilon$. Weights for prior $p(\boldsymbol{\pi})$: $\alpha_0$.
Output: Cluster assignment $q^*(\mathbf{Z}) \approx p(\mathbf{Z}|\mathbf{S}, \theta_{opt})$.
1  Initialize $\{\theta_k\}_{k=1}^K$ for $K$ NTPPs. Initialize
     $\{\alpha_k = \frac{\alpha_0 + N}{K}\}_{k=1}^K$ for $q(\boldsymbol{\pi})$;
2  Pre-train each NTPP on $\mathbf{S}$ to prevent bad initialization;
3  for $epoch = 1, \cdots, T$ do
4      for $iter = 1, \cdots, \lceil \frac{N}{B} \rceil$ do
5          Sample a batch of sequences $\mathbf{S}'$ from $\mathbf{S}$;
           // E-step: update $q(\mathbf{Z}')$
6          $\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] = \psi(\alpha_k) - \psi(\sum_{k=1}^K \alpha_k)$
7          $\log p(S_n|\theta_k) = \log[\text{NTPP}(S_n|\theta_k)] \quad S_n \in \mathbf{S}'$
8          $r_{nk} = \frac{\exp\left(\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] + \log p(S_n|\theta_k)\right)}{\sum_{k=1}^K \exp\left(\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] + \log p(S_n|\theta_k)\right)}$
           // E-step: update $q(\boldsymbol{\pi})$
9          $\alpha_k = (1-\eta)\alpha_k + \eta(\alpha_0 + \frac{N}{|\mathbf{S}'|}\sum_{S_n \in \mathbf{S}'} r_{nk})$
           // M-step: update $\theta$
10         for $step = 1, \cdots, M$ do
11             $\widetilde{\mathcal{L}}(\theta) = \sum_{S_n \in \mathbf{S}'} \sum_{k=1}^K r_{nk} \log p(S_n|\theta_k)$;
12             Update $\theta$ using SGD with $-\widetilde{\mathcal{L}}(\theta)$ as loss;
       // Remove redundant clusters
13     $\forall k$, remove $\theta_k$ and distribute $\alpha_k$ to other clusters
         equally if $\alpha_k < \epsilon$;
14 Compute $r_{nk}, n \in \{1, \ldots, N\}, k \in \{1, \ldots, K\}$ using
     the trained model (line 6 to 8);
15 Return $q^*(z_{nk} = 1) = r_{nk}$;
```

4 Experiments

4.1 Protocols

We conduct experiments on 4 synthetic datasets and 3 real-world datasets to evaluate the performance of NTPP-MIX.

Synthetic datasets We generate event sequences with 5 different processes: **Homo** Homogeneous Possion process. **In-homo** In-homogeneous Possion process. **Inhibit** Process in which events of different types inhibit each other. **Excite** Process in which events excite each other (Hawkes). **In&Ex** Process in which relation is either inhibition or excitation. Each process generates 4,000 event sequences, and each sequence contains $L_n = 50$ events with number of event types $C = 5$.We merge above sequences to generate 4 datasets with different ground truth number of clusters (K_{GT}):

$\mathbf{K_{GT} = 2}$: Homo + In-homo;
$\mathbf{K_{GT} = 3}$: Homo + In-homo + Inhibit;
$\mathbf{K_{GT} = 4}$: Homo + In-homo + Inhibit + Excite;
$\mathbf{K_{GT} = 5}$: Homo + In-homo + Inhibit + Excite + In&Ex.

Real-world datasets We use 3 real-world datasets:
1) Stock[2] contains daily stock prices of 31 companies. We extract three types of event: 'up', 'down', 'unchanged' from each stock. We further partition sequences by every season and obtain 1,488 sequences with average length of 49.
2) eCommerce[3] contains users' behavior in a cosmetics online store. Each user's behaviors are categorized into four types: 'view', 'cart', 'remove-from-cart', 'purchase'. This dataset contains 6,200 sequences with average length of 64.
3) Neuron[4] contains spike record of 219 M1 neurons. Every time a neuron spikes is recorded as an event, these events form a uni-dimensional event sequence. This dataset contains

[2]www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231
[3]www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop
[4]www.kordinglab.com/spykes/index.html

Dataset	$K_{GT}=2$		$K_{GT}=3$		$K_{GT}=4$		$K_{GT}=5$	
Metric	CP	ARI	CP	ARI	CP	ARI	CP	ARI
HKS+BGM	.5818 ± .0805	.0526 ± .0852	.4892 ± .0261	.0893 ± .0424	.4207 ± .0320	.0787 ± .0258	.4795 ± .0037	.1923 ± .0331
ADM4+BGM	.6476 ± .0812	.1134 ± .0851	.4567 ± .0131	.0544 ± .0128	.3896 ± .0135	.0465 ± .0074	.4801 ± .0064	.1979 ± .0209
NPHC+BGM	.5207 ± .0164	.0027 ± .0023	.4383 ± .0058	.0378 ± .0044	.3517 ± .0146	.0384 ± .0108	.3222 ± .0037	.0441 ± .0023
DIS+SC	.9534 ± .0000	.8222 ± .0000	.9407 ± .0000	.8333 ± .0000	.2557 ± .0000	.0001 ± .0000	.2225 ± .0000	.0018 ± .0000
DMHP	.9598 ± .0007	.8455 ± .0026	.8789 ± .1069	.7473 ± .1327	.5292 ± .0423	.2730 ± .0542	.6624 ± .0905	.4753 ± .1142
RMTPP-MIX	**.9898** ± .0004	**.9597** ± .0015	**.9930** ± .0006	**.9793** ± .0018	.9532 ± .0656	.9097 ± .1106	.9273 ± .0748	.8844 ± .0885
SAHP-MIX	**.9898** ± .0019	.9594 ± .0073	.9891 ± .0008	.9677 ± .0023	.9840 ± .0008	.9578 ± .0020	.9686 ± .0109	.9248 ± .0241
LNM-MIX	.9882 ± .0020	.9535 ± .0080	.9928 ± .0008	.9785 ± .0023	**.9873** ± .0004	**.9666** ± .0010	**.9867** ± .0011	**.9673** ± .0026

Table 1: CP and ARI on synthetic datasets. We run 5 trials by random initialization and report the mean and standard deviation. Our MIX framework can readily incorporate existing modules: RMTPP [Du *et al.*, 2016], SAHP [Zhang *et al.*, 2020], and LNM [Shchur *et al.*, 2020].

3,718 sequences with average length of 265.

Models for comparison. We compare the following clustering methods on both synthetic and real-world datasets:
1) HKS+BGM. Learn a specific Hawkes process for each event sequence and apply Bayesian Gaussian Mixture (BGM) model to learned parameters for clustering.
2) ADM4+BGM. Learn a specific ADM4 [Zhou *et al.*, 2013] for each sequence and apply BGM to the learned parameters.
3) NPHC+BGM. Learn Non Parametric Hawkes Cumulant (NPHC) [Xu *et al.*, 2016] for each sequence and apply BGM.
4) DIS+SC. Define distance for event sequence [Iwayama *et al.*, 2017], apply Spectral Clustering to the distance matrix.
5) DMHP [Xu and Zha, 2017]. This is the most related method to our work, which mixes several Hawkes processes to generate a Dirichlet Mixture Model of Hawkes Processes.
6) RMTPP-MIX. Our NTPP-MIX with RMTPP [Du *et al.*, 2016] as components. RMTPP uses RNN to model intensity.
7) SAHP-MIX. Our NTPP-MIX with SAHP [Zhang *et al.*, 2020] as components. It uses Transformer to model intensity.
8) LNM-MIX. Our NTPP-MIX with LNM [Shchur *et al.*, 2020] as components. LNM is intensity-free and MLE-based.

The first four methods are two-step pipelines which extract features and use them for clustering. DMHP and three NTPP-MIXs are joint models which perform clustering directly.

We implement all three versions of NTPP-MIX using PyTorch. On all datasets, hyper-parameters in Algorithm 1 are set as: $B = 128, T = 100, M = 5, \alpha_0 = \frac{1}{K}$, pre-train epoch is set to 5. On synthetic datasets, $\eta = 0.01, \epsilon = \frac{N}{5*K}$, Adam with $lr = 0.005$ is used for SGD; On real-world datasets, $\eta = \frac{B}{N}, \epsilon = \frac{N}{10*K}$, Adam with $lr = 0.0005$ is used.

Clustering metrics. We use three metrics: the first two for synthetic datasets following [Xu and Zha, 2017; Wu *et al.*, 2022], the third for real-world datasets where the ground truth assignment is unknown following [Zhang and Yan, 2021].
1) Clustering Purity (CP). Ratio of correct assignments.
2) Adjusted Rand index (ARI). Similarity of learned and ground truth assignments [Nguyen *et al.*, 2010].
3) Expected Log Likelihood (ELL). The expected log likelihood w.r.t. cluster assignment on testing set, i.e. we randomly split the data set into training and testing sets, train the model on training set, fix $q(\boldsymbol{\pi}), \theta$ and estimate the cluster assignments $q(\mathbf{Z}_{test})$ for testing set, then report $\mathbb{E}_{q(\mathbf{Z}_{test})}[\log p(\mathbf{S}_{test}|\mathbf{Z}_{test}, \theta)]$. A model can get higher ELL on testing set by recovering the real distribution.

Ground Truth K_{GT}	2	3	4	5
DMHP	3.5±0.50	4.9±0.30	2.8±0.75	4.4±0.92
RMTPP-MIX	**2.4**±0.49	3.2±0.40	4.3±0.46	**5.0**±0.78
SAHP-MIX	2.5±0.50	3.3±0.46	**4.0**±0.00	6.0±0.45
LNM-MIX	2.7±0.64	**3.1**±0.30	4.4±0.66	5.3±0.64

Table 2: Number of clusters selected by DMHP and NTPP-MIX on synthetic datasets. For each model, we run 10 trials.

Dataset	Stock	eCommerce	Neuron
HKS+BGM	-132.33	-229.77	-69.75
ADM4+BGM	-181.09	-184.70	-51.73
DMHP	-97.35	-158.05	25.47
RMTPP-MIX	-96.15	N/A	38.72
SAHP-MIX	-93.23	-106.09	68.01
LNM-MIX	**-77.27**	**-97.31**	**75.89**

Table 3: ELL estimation on real-world datasets. NPHC+BGM and DIS+SC are MLE-free methods. Time intervals in eCommerce vary from seconds to hours, causing overflow in RMTPP (denoted N/A).

4.2 Experiments on Synthetic Datasets

We first assume the ground truth cluster number K_{GT} is given and assign $K = K_{GT}$ for each method. Table 1 shows the CP and ARI. In general, joint models (DMHP and NTPP-MIX) outperform two-step pipelines. As feature extraction and clustering of two-step models are relatively independent, the extracted feature may not be suitable for clustering. Performance of DIS+SC is satisfactory on $K_{GT} = 2, 3$ but drops sharply on $K_{GT} = 4, 5$. This shows it is hard to define a universal distance for typed event sequences.

For joint models, our NTPP-MIXs constantly outperform DMHP, no matter what NTPP we use. The mixture components in DMHP are Hawkes processes, i.e. different clusters are defined as Hawkes processes with different parameters. This limits model capacity as dynamic behind data can be much more complex. Moreover, when we add sequences generated by Hawkes processes in $K_{GT} = 4$, performance of DMHP drops suddenly and the clustering result collapses to one cluster. The mixture components of our NTPP-MIXs are neural-network-based NTPPs, which are more general and flexible. The standard deviation also shows our NTPP-MIXs are more stable than DMHP. We further visualize similarity matrices of some models on $K_{GT} = 5$ in Fig. 2. It can be

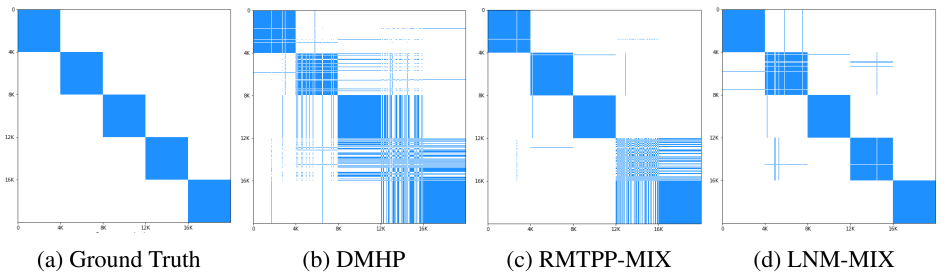
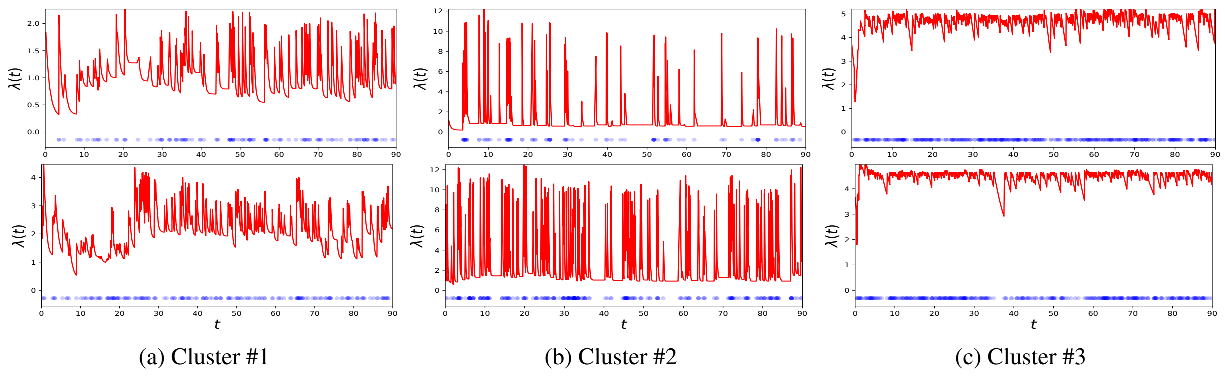

(a) Ground Truth (b) DMHP (c) RMTPP-MIX (d) LNM-MIX

Figure 2: Binary similarity on synthetic dataset $K_{GT} = 5$: (i, j) is colored blue if sequence i and j are in the same cluster.



(a) Cluster #1 (b) Cluster #2 (c) Cluster #3

Figure 3: Clustered sequences (blue dots) and corresponding intensities (red lines) of dataset Neuron using SAHP-MIX.

seen that result of LNM-MIX is the closest to ground truth.

We further compare NTPP-MIXs' ability to choose the appropriate number of clusters with DMHP. We set the initial cluster number $K = 2K_{GT}$, and run 10 trials with different random initialization. The selected number of clusters is shown in Table 2. Note that number of clusters selected by our NTPP-MIX is closer to ground truth and more stable.

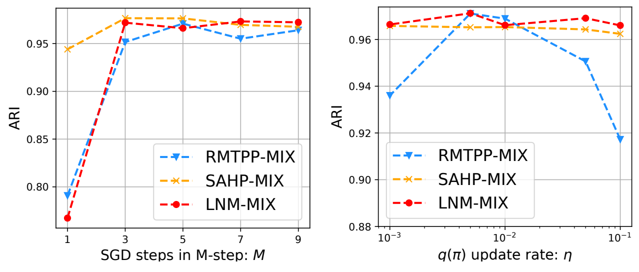
4.3 Experiments on Real-world Datasets

For real-world datasets, we randomly sample 60% sequences for training and use the other 40% for testing. Table 3 reports the ELL on testing set, which shows that NTPP-MIXs outperform baselines by a large margin. Two-step models perform poorly and are not as interpretable as joint models: it is hard to explain what we get by directly clustering parameters of Hawkes processes. DMHP also performs not very well due to the strong parametric assumption. Moreover, DMHP collapses on Neuron and assigns all sequences to one cluster.

Fig. 3 shows clustered sequences and corresponding intensities of Neuron using SAHP-MIX: in Cluster #1, when an event occurs, the intensity increases first then decreases like traditional Hawkes process. In Cluster #2, the intensity has a small constant base and spikes sharply when an event occurs. In Cluster #3, intensity decreases when no event occurs, while a new event will pull the intensity back to the baseline.

4.4 Parameter Sensitivity Study

We perform sensitivity analysis of our training algorithm on Dataset $K_{GT} = 5$. **Number of SGD steps in M-step** (M in Eq. 15): In Fig. 4(a), when $M = 1$ all three NTPP-MIXs perform not so well, as we have stated in Eq. 15 that it needs multiple SGD steps. When $M \geq 3$, all models converge and get close performance. **$\mathbf{q(\boldsymbol{\pi})}$ update rate** (η in Eq. 16): η controls how much update is made to $q(\boldsymbol{\pi})$ with a batch of data. We range η from 0.001 to 0.1 and get Fig. 4(b). RMTPP-MIX performs not so well when η is too large or small, while SAHP-MIX and LNM-MIX are stable w.r.t η.



(a) # SGD steps in M-step (b) $q(\boldsymbol{\pi})$ update rate

Figure 4: The sensitivity of two parameters in NTPP-MIX.

5 Conclusion

We have proposed NTPP-MIX, a framework that can utilize existing NTPPs for multi-dimensional event sequence clustering. NTPP-MIX can be efficiently trained by our devised combination of variational EM algorithm and SGD. This is the first neural model for multi-dimensional events clustering, and it is more flexible than traditional parametric models. Experiments show the effectiveness of NTPP-MIX in terms of fitting complex dynamics of event sequence, against state-of-the-art methods on public datasets.

References

[Bishop and Nasrabadi, 2007] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern Recognition and Machine Learning*. J. Electronic Imaging, 2007.

[Chen *et al.*, 2018] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[Chen *et al.*, 2021] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. In *ICLR*, 2021.

[Daley and David, 2007] D.J. Daley and Vere-Jones David. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Springer Science & Business Media, 2007.

[Du *et al.*, 2016] Nan Du, H. Dai, Rakshit S. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, 2016.

[Enguehard *et al.*, 2020] Joseph Enguehard, Dan Busbridge, Adam James Bozson, Claire Woodcock, and Nils Y. Hammerla. Neural temporal point processes for modelling electronic health records. In *ML4H*, 2020.

[Hawkes, 1971] Alan G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971.

[Hoffman *et al.*, 2013] Matthew D. Hoffman, David M. Blei, Wang Chong, and John Paisley. Stochastic variational inference. *JMLR*, 14, 2013.

[Iwayama *et al.*, 2017] Koji Iwayama, Yoshito Hirata, and Kazuyuki Aihara. Definition of distance for nonlinear time series analysis of marked point process data. *Physics Letters A*, 2017.

[Jia and Benson, 2019] Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. In *NeurIPS*, 2019.

[Li *et al.*, 2018] Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning temporal point processes via reinforcement learning. In *NeurIPS*, 2018.

[Mei and Eisner, 2016] Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*, 2016.

[Nguyen *et al.*, 2010] Xuan Vinh Nguyen, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *JMLR*, 2010.

[Omi *et al.*, 2019] Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *NeurIPS*, 2019.

[Opper and Saad, 2001] Manfred Opper and David Saad. *Advanced mean field methods: theory and practice*. MIT press, 2001.

[Sharma *et al.*, 2021] Karishma Sharma, Yizhou Zhang, Emilio Ferrara, and Yan Liu. Identifying coordinated accounts on social media through hidden influence and group behaviours. *KDD*, 2021.

[Shchur *et al.*, 2020] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *ICLR*, 2020.

[Sutskever *et al.*, 2013] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[Upadhyay *et al.*, 2018] U. Upadhyay, Abir De, and Manuel Gomez-Rodriguez. Deep reinforcement learning of marked temporal point processes. In *NeurIPS*, 2018.

[Wu *et al.*, 2022] Weichang Wu, Junchi Yan, Xiaokang Yang, and Hongyuan Zha. Discovering temporal patterns for event sequence clustering via policy mixture model. *TKDE*, 2022.

[Xiao *et al.*, 2017] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Xiaokang Yang, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *NeurIPS*, 2017.

[Xu and Zha, 2017] Hongteng Xu and Hongyuan Zha. A dirichlet mixture model of hawkes processes for event sequence clustering. In *NeurIPS*, 2017.

[Xu *et al.*, 2014] Lizhen Xu, Jason A. Duan, and Andrew Whinston. Path to purchase: A mutually exciting point process model for online advertising and conversion. *Management Science*, 2014.

[Xu *et al.*, 2016] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. Learning granger causality for hawkes processes. In *ICML*, 2016.

[Yan *et al.*, 2016] Junchi Yan, Shuai Xiao, Changsheng Li, Bo Jin, Xiangfeng Wang, Bin Ke, Xiaokang Yang, and Hongyuan Zha. Modeling contagious merger and acquisition via point processes with a profile regression prior. In *IJCAI*, 2016.

[Yan *et al.*, 2018] Junchi Yan, Xin Liu, Liangliang Shi, Changsheng Li, and Hongyuan Zha. Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning. In *IJCAI*, 2018.

[Zhang and Yan, 2021] Yunhao Zhang and Junchi Yan. Neural relation inference for multi-dimensional temporal point processes via message passing graph. In *IJCAI*, 2021.

[Zhang *et al.*, 2020] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *ICML*, 2020.

[Zhou *et al.*, 2013] K. Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multidimensional hawkes processes. In *AISTATS*, 2013.

[Zuo *et al.*, 2020] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *ICML*, 2020.