
Variational Inference for Neyman-Scott Processes

Chengkuan Hong
Tsinghua University
chong009@ucr.edu

Christian R. Shelton
University of California, Riverside
cshelton@cs.ucr.edu

Abstract

Neyman-Scott processes (NSPs) have been applied across a range of fields to model points or temporal events with a hierarchy of clusters. Markov chain Monte Carlo (MCMC) is typically used for posterior sampling in the model. However, MCMC’s mixing time can cause the resulting inference to be slow, and thereby slow down model learning and prediction. We develop the first variational inference (VI) algorithm for NSPs, and give two examples of suitable variational posterior point process distributions. Our method minimizes the inclusive Kullback-Leibler (KL) divergence for VI to obtain the variational parameters. We generate samples from the approximate posterior point processes much faster than MCMC, as we can directly estimate the approximate posterior point processes without any MCMC steps or gradient descent. We include synthetic and real-world data experiments that demonstrate our VI algorithm achieves better prediction performance than MCMC when computational time is limited.

1 INTRODUCTION

Neyman-Scott processes (Neyman and Scott, 1958) have achieved great success in many fields, *e.g.*, pandemic modeling (Park et al., 2022), neuroscience (Williams et al., 2020), and seismology (Hong and Shelton, 2022), where posterior sampling plays an important role. Many MCMC algorithms (*e.g.*, Møller and Waagepetersen, 2003; Williams et al., 2020; Hong and Shelton, 2022) have been proposed for the posterior sampling. However, a large number of mixing steps is generally required for MCMC, which makes it inefficient to generate samples from the posterior point process.

To accelerate the approximate posterior inference of the hidden points, we propose a variational inference (VI) algorithm. We employ an autoencoder to construct two variational posterior point processes. One of these has the same functional form as the virtual point processes, which work as auxiliary variables for MCMC, in Hong and Shelton (2022), and the other uses self-attention (Vaswani et al., 2017) to construct a more general approximation. We can sample from our approximation much faster than from MCMC because mixing is not needed for the approximation. With faster sampling from our approximate posterior point processes, we are able to achieve better prediction results when there is a time constraint.

We briefly review VI and Neyman-Scott processes in Sec. 2. Sec. 3 gives a general formula for our approximate posterior point processes, and Sec. 4 gives two specific examples for the approximation. Sec. 5 gives details about the inference procedure. Sec. 6 explains how to do prediction and discusses why our approximate posterior point processes can behave better than MCMC when there is a constraint for the time. Finally, we use experiments to support our claim in Sec. 7 and conclude this paper in Sec. 8. While our experiments are only for temporal point processes, our algorithms can be applied to a general spatio-temporal point process as explained in Remark 1.

2 BACKGROUND

2.1 Variational Inference

VI transforms a posterior inference problem into an optimization problem, and it has already been very successful for probabilistic modeling. Usually, VI first constructs a family of approximate distributions q , and then adjusts the parameters of q to approach the true posterior distribution p by minimizing a divergence metric. Most VI algorithms try to minimize the exclusive KL divergence, $\text{KL}(q \parallel p)$ (*e.g.*, Jordan et al., 1999; Kingma and Welling, 2014; Ranganath et al., 2014; Blei et al., 2017). While it is computationally efficient to optimize the exclusive KL divergence, it can lead to underestimation of the uncertainty of the posterior (Naesseth et al., 2020). To mitigate the underestimation issue, Naesseth et al. (2020) proposed a VI algorithm, called

Markovian score climbing (MSC), that minimizes the inclusive KL divergence, $\text{KL}(p \parallel q)$. MSC uses MCMC to get an unbiased estimate of the gradient of the inclusive KL divergence. Naesseth et al. (2020) also shows that MSC can be combined with maximum likelihood estimation (MLE) to jointly learn variational parameters and model parameters.

In a similar fashion, we design a VI algorithm for NSPs that minimizes inclusive KL divergence. Different from MSC, whose variational parameters are for Markov kernels, our variational parameters are for auxiliary variables of MCMC. Moreover, the number of dimensions is fixed in Naesseth et al. (2020), and our MCMC has an unbounded number of dimensions. To the best of our knowledge, we are the first to design a VI algorithm for NSPs and there does not exist any pre-existing work for minimizing the exclusive KL divergence. While we only focus on the VI for minimizing inclusive KL divergence in this paper, the minimization of exclusive KL divergence could also be interesting and we leave it for future research.

2.2 Neyman-Scott Processes

The Neyman-Scott process (NSP) was first proposed by Neyman and Scott (1958). NSPs are a class of hierarchical point process models built by stacking Poisson processes into a network. NSPs were originally univariate, *i.e.*, can only be applied to events with one type or mark. Hong and Shelton (2022) introduce multivariate NSPs with hierarchical structures, called deep Neyman-Scott processes (DNSPs). We use “deep” to differentiate our work from other work that only has one hidden layer (*e.g.*, Møller and Waagepetersen, 2003; Linderman et al., 2017; Williams et al., 2020). While Hong and Shelton (2022) only have experiments for temporal point processes (TPPs), their algorithms can be used for spatio-temporal NSPs with multiple dimensions.

The major assumption of DNSPs is that the observed events (points) are triggered by the hidden events (points). For example, earthquakes are usually triggered by sudden energy releases of the Earth. The times and locations of energy releases are hidden (not directly observed) points. We can leverage the power of DNSPs to infer the potential times and locations of energy releases given the times, locations, and magnitudes of the observed earthquakes. Then, based on the inferred hidden information, we can predict the time and the magnitude of the next future earthquake conditional on the observed earthquakes.

More generally, the hierarchical systems of DNSPs are formed by the iterative generation procedures of the hidden points. Each hidden point (parent) can trigger a set of hidden points as its children. The generated child points can also work as parent points and trigger their own sets of children. This iterative process continues until the observed

points are generated. With this hierarchical structure, we are able to infer more hidden information. For example, we can infer the hidden triggers of the energy releases of earthquakes. The experiments in Hong and Shelton (2022) and our experiments in Sec. 7.2 also show that more hidden point processes can bring better prediction performance.

We typically use conditional intensity functions (CIFs) to describe TPPs.

Definition 1 (conditional intensity function). *The conditional intensity function (CIF) of a TPP is defined as*

$$\lambda(t) = \lim_{\Delta t \rightarrow 0} \frac{\Pr(\text{One event in } [t, t + \Delta t] \mid \mathcal{H}_t)}{\Delta t},$$

where \mathcal{H}_t represents the events that happened before t .

The log-likelihood function for a TPP defined on $(0, T]$ is

$$\sum_{i=1}^n \log \lambda(t_i) - \int_0^T \lambda(t) dt,$$

where $\lambda(t)$ is the CIF and $t_1 \leq t_2 \leq t_3 \leq \dots \leq t_n$ are the times of observations.

Figs. 1 and 2 in Hong and Shelton (2022) provide good illustrations for deep Neyman-Scott processes, where each observed sequence of events has L hidden layers of (multiple) Poisson processes. The CIF for each Poisson process of layer l is determined by the events from the layer immediately above layer $l + 1$. For simplicity, we omit the data index n in the following formal description of the generative process for DNSPs, and only consider TPPs in this paper. Let $Z_{\ell,k}$ represent the k -th hidden Poisson process at layer ℓ , K_ℓ be the number of point processes at layer ℓ , $\phi_{\theta_{(\ell+1,i) \rightarrow (\ell,k)}}$ be the kernel function that connects $Z_{\ell,k}$ and $Z_{\ell+1,i}$, $\mathbf{z}_{\ell,i} = \{t_{\ell,i,j}\}_{j=1}^{m_{\ell,i}}$ be a realization for $Z_{\ell,i}$, $\mathbf{z}_\ell = \{\mathbf{z}_{\ell,i}\}_{i=1}^{K_\ell}$, and $\mathbf{z}_0 = \mathbf{x}$. To generate the events for the bottom layer, we first need to generate random events from the top layer. The CIF for $Z_{L,k}$ is

$$\lambda_{L,k}(t) = \mu_k, \text{ where } \mu_k > 0.$$

Then, we can draw samples for each hidden Poisson process conditional on the Poisson processes that are from the layers immediately above. The CIF for $Z_{\ell,k}$ is

$$\lambda_{\ell,k}(t) = \sum_{i=1}^{K_{\ell+1}} \sum_{j=1}^{m_{\ell+1,i,j}} \phi_{\theta_{(\ell+1,i) \rightarrow (\ell,k)}}(t - t_{\ell+1,i,j}). \quad (1)$$

We denote this point process as the real point process (RPP), to make a distinction with the virtual point process (VPP), below, used to add auxiliary variables to a Markov chain.

2.3 Kernel Function

We choose the kernel function to be a Weibull kernel,

$$\phi_{\theta}(x) = \begin{cases} p \cdot \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & \text{for } x > 0, k, \lambda > 0, \\ 0 & \text{for } x \leq 0, \end{cases}$$

where $\theta = \{p, k, \lambda\}$. Here, we omit the subscripts of θ in the definition for simplicity as it is the general form. Similar to the gamma kernel used in previous work, the Weibull kernel converges to 0 when x goes to infinity as the influence of the events from the past will fade eventually. The shape of a Weibull kernel is very similar to a gamma kernel, and, with different combinations of the parameters, the Weibull kernel can also monotonically decrease or behave like a Gaussian function. But, the Weibull kernel has the advantage over the gamma kernel that the gradients of the Weibull kernel function itself and the integral of the Weibull kernel function are both analytically available. The only restriction for the kernel function is that Eqs. 1, 2, 5, and 6 are non-negative.

2.4 Monte Carlo Expectation-Maximization

Hong and Shelton (2022) introduced a Monte Carlo expectation-maximization (MCEM) algorithm for inference.

Virtual events work as auxiliary variables to accelerate the mixing of MCMC chains, and are candidate events for real events. When searching for the positions of the real events, we only need to search where the virtual events appear instead of in the whole space. Because of this, the virtual point processes (VPPs) $\tilde{\mathbf{Z}}$ are designed to be as close as possible to the true posterior point processes of the real point processes \mathbf{Z} . For the same reason, the CIFs for VPPs evolve in the reverse directions (temporally backward and “up” the layers) relative to the real point processes (temporally forward and “down” the layers).

Similar to the RPPs, $\tilde{Z}_{\ell,k}$ represents the k -th hidden Poisson process on layer ℓ , $\tilde{\phi}_{\tilde{\theta}_{(\ell-1,i) \rightarrow (\ell,k)}}$ is the kernel function connects $Z_{\ell-1,i}$ and $\tilde{Z}_{\ell,k}$, and $\tilde{\mu}_{\ell,k} \geq 0$ is the base rate. In this case, the history \mathcal{H}_t in Definition 1 becomes the events that happened after t . Then, the CIF for $\tilde{Z}_{\ell,k}$ conditioned on $\mathbf{Z}_{\ell-1}$ is

$$\tilde{\lambda}_{\ell,k}(\tilde{t}) = \tilde{\mu}_{\ell,k} + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}} \tilde{\phi}_{\tilde{\theta}_{(\ell-1,i) \rightarrow (\ell,k)}}(t_{\ell-1,i,j} - \tilde{t}). \quad (2)$$

We briefly outline the MCEM algorithm (Hong and Shelton, 2022) in Alg. 1. $\log p(\mathbf{x}, \mathbf{z})$ in line 3 is the joint log-likelihood of the hidden RPPs and the observation \mathbf{x} . $\log q(\mathbf{z})$ in line 4 represents the log-likelihood for the VPPs. There is a slight difference between Alg. 1 and the employed MCEM algorithm: We directly maximize the

Algorithm 1 MCEM for DNSPs

Input: data \mathbf{x} , model \mathcal{M}

Initialization: parameters for RPPs Θ_0 , parameters for VPPs $\tilde{\Theta}_0$, initial sample for RPPs \mathbf{z}_0 , and iterations N .

Output: $\Theta_N \approx \Theta^*$, $\tilde{\Theta}_N \approx \tilde{\Theta}^*$

- 1: **for** $n = 1$ **to** N **do**
 - 2: sample $\mathbf{z}_n \sim p(\mathbf{z} | \mathbf{x}; \Theta_{n-1}, \mathbf{z}_{n-1})$
 - 3: $\Theta_n \leftarrow \Theta_{n-1} + \eta_n \nabla_{\Theta} \log p(\mathbf{x}, \mathbf{z}; \Theta_{n-1})$
 - 4: $\tilde{\Theta}_n \leftarrow \tilde{\Theta}_{n-1} + \tilde{\eta}_n \nabla_{\tilde{\Theta}} \log q(\mathbf{z}; \mathbf{x}, \tilde{\Theta}_{n-1})$
 - 5: **end for**
-

constant rates μ instead of doing gradient ascent. Line 2 does posterior sampling via MCMC, and the previous sample \mathbf{z}_{n-1} serves as the initial state of the next MCMC step. Line 3 is used to maximize the marginal likelihood. Line 4 adjusts the parameters for VPPs to make VPPs closer to the posterior RPPs. η_n and $\tilde{\eta}_n$ are step sizes for gradient ascent.

Hong and Shelton (2022) only use Alg. 1 to learn the model parameters for DNSPs, without any discussion of the variational inference. However, we can directly use Alg. 1 to learn the parameters for our approximate point processes, which is one of our main contributions.

3 APPROXIMATE POSTERIOR POINT PROCESSES

We denote our variational approximate point processes as \mathbf{Z}^I , and we assume they have hierarchical structures like DNSPs. However, the approximate point processes are generated from bottom to top (upward), instead of from top to bottom (downward). The upward approximation mechanism allows us to infer the approximate posterior distributions of the point processes directly from the observation, which can help accelerate the sampling process for the posterior point processes. Moreover, the events from the layers immediately below can give heuristics for the positions of the posterior events from the layers immediately above.

Generative Semantics for \mathbf{Z}^I The approximate posterior point processes are generative models, and they are denoted as $\mathbf{Z}^I = \{\mathbf{Z}_1^I, \dots, \mathbf{Z}_L^I\}$, with $\mathbf{Z}_\ell^I = \{Z_{\ell,k}^I\}_{k=1}^{K_\ell}$.

\mathbf{Z}_1^I is assumed to be Poisson processes, and the CIF for $\mathbf{Z}_{1,k}^I$ is

$$\lambda_{1,k}^I(t) = q_{1,k}^I(t; \mathbf{z}_0 = \mathbf{x}, \theta_{1,k}^I(t)), \quad (3)$$

where $q_{1,k}^I(t; \mathbf{x}, \theta_{1,k}^I(t))$ is a function of t with some parameters determined by the events from the observation \mathbf{x} , and other parameters $\theta_{1,k}^I(t)$ that can depend on time t .

Next, we draw samples for each \mathbf{Z}_ℓ^I conditional on the samples $\mathbf{z}_{\ell-1}^I$ from $\mathbf{Z}_{\ell-1}^I$. The CIF for $\mathbf{Z}_{\ell,k}^I$ conditional on $\mathbf{Z}_{\ell-1}^I$

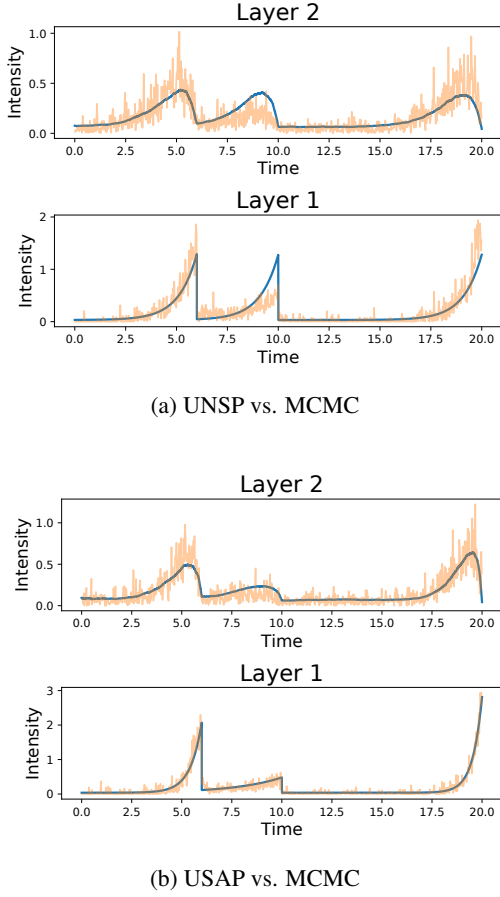


Figure 1: Comparison Between UNSP and USAP

is

$$\lambda_{\ell,k}^I(t) = q_{\ell,k}^I(t; \mathbf{z}_{\ell-1}^I, \boldsymbol{\theta}_{\ell,k}^I(t)), \quad (4)$$

where $q_{\ell,k}^I(t; \mathbf{z}_{\ell-1}^I, \boldsymbol{\theta}_{\ell,k}^I(t))$ is a function of t with some parameters determined by the events from $\mathbf{z}_{\ell-1}^I$, and other parameters can also depend on time t .

4 EXAMPLES FOR VARIATIONAL POINT PROCESSES

As explained above, there are two major goals when constructing the approximations: (1) the approximations should be able to propagate the information from the observations to the top, and (2) the approximate posterior point processes should be close to the true posterior point processes. For these purposes, we consider the following two examples of functional forms for $q_{\ell,k}^I(\cdot)$ in this paper: upward Neyman-Scott processes and upward self-attention processes.

Upward Neyman-Scott Processes (UNSPs) Like the virtual point processes, we can assume the approximate point processes are NSPs evolving in an upward direction.

In this case, the CIF is

$$\begin{aligned} \lambda_{\ell,k}^I(t) &= q_{\ell,k}^I(t; \mathbf{z}_{\ell-1}^I, \boldsymbol{\theta}_{\ell,k}^I) \\ &= \mu_{\ell,k}^I + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}} \phi_{\boldsymbol{\theta}_{\ell,k}^I} (t_{\ell-1,i,j} - t), \end{aligned} \quad (5)$$

where $\boldsymbol{\Theta}_{\ell,k}^I = \left\{ \mu_{\ell,k}^I, \left\{ \boldsymbol{\theta}_{\ell,k}^I \right\}_{i=1}^{K_{\ell-1}} \right\}$ and $\mu_{\ell,k}^I \geq 0$. We see that Eq. 5 has the same functional form as Eq. 2.

Upward Self-Attention Processes (USAPs) Self-attention has been widely used in the modeling for point processes (e.g., Zuo et al., 2020; Zhang et al., 2020; Chen et al., 2021). Here, we use self-attention to encode the event information from a layer below to a layer above. Each event $t_{\ell-1,i,j}$ is encoded into a hidden vector $\mathbf{h}_{\ell-1,i,j} = f_{\text{Attn}}(t_{\ell-1,i,j}, \mathbf{z}_{\ell-1}^I = \{\{t_{\ell-1,i,j}\}_{j=1}^{m_{\ell-1,i}}\}_{i=1}^{K_{\ell-1}})$ through self-attention. The CIF for $t_{\ell-1,i,j-1} < t \leq t_{\ell-1,i,j}$ becomes

$$\begin{aligned} \lambda_{\ell,k}^I(t) &= q_{\ell,k}^I(t; \mathbf{z}_{\ell-1}^I, \boldsymbol{\theta}_{\ell,k}^I) \\ &= \mu_{\ell,k}^I + \phi_{\boldsymbol{\theta}_{\ell,k}^I} (t_{\ell-1,i,j} - t), \end{aligned} \quad (6)$$

where $\boldsymbol{\Theta}_{\ell,k}^I = \left\{ \mu_{\ell,k}^I, \left\{ \left\{ \boldsymbol{\theta}_{\ell,k}^I \right\}_{i=1}^{K_{\ell-1}} \right\}_{j=1}^{m_{\ell-1,i}} \right\}$, $\mu_{\ell,k}^I \geq 0$, and $\boldsymbol{\theta}_{\ell,k}^I$ is the output of a linear transformation of the hidden vector $\mathbf{h}_{\ell-1,i,j}$. More details can be found in Appx. A.

Comparison We plot the intensity functions estimated by using UNSPs, USAPs, and MCMC in Fig. 1 to compare the approximation abilities of UNSPs and USAPs. We choose 3 events by hand in the observation (times at 6, 10, and 20) to illustrate the VI effects in a clean and typical example, and we use MCMC, UNSPs, and USAPs to infer the posterior point processes.

The model parameters for the underlying DNSP are fixed. We only adjust the variational parameters to let the variational point processes (USAPs and UNSPs) approximate the posterior point processes. Layer 2 is further away from the observation than layer 1. The yellow lines with many spikes are the approximate intensity functions estimated by using the samples from MCMC. We divide the time interval into many small subintervals, and for each small subinterval, the approximate intensity function is the number of events in that small subinterval divided by the length of that small subinterval. The blue solid lines are approximate intensity functions for VI. We obtain the intensity function for layer 1 directly from $q_{1,0}^I(\cdot)$. For the approximate intensity function for layer 2, we first generate many samples from $q_{1,0}^I(\cdot)$ which induce samples of $q_{2,0}^I(\cdot)$. Then the approximate intensity function for layer 2 is the mean of all the samples for $q_{2,0}^I(\cdot)$. We see that USAP fits the approximate intensity functions estimated by the MCMC better than UNSP.

The better approximation of USAPs in Fig. 1 comes from the fact that the kernel function can adjust its parameters at each interval independently. While for UNSPs, the CIF at each interval is affected by all the kernel functions that are triggered by the events which happen after that interval and are from the layer immediately below.

In addition, for the CIFs of the approximate posterior point processes at time t , USAPs can capture the information from both the events happening before t and the events happening after t , while UNSPs can only propagate the information from the future to the past. Both the events happening before t and the events happening after t have an influence on the posterior point processes at time t (Hong and Shelton, 2022, Proposition 1). Therefore, USAPs are better than UNSPs, since UNSPs only include the information of the events happening after t .

Algorithm 2 Prediction with MCMC or approximation

Input: observed data $\mathbf{x} = \{e_1, e_2, \dots, e_n\}$, where $e_i = (t_i, k_i)$, sampler $G(\mathbf{x}, \Theta^G)$, and model \mathcal{M} .

Initialization: Θ , $\tilde{\Theta}$, and sample size \mathcal{S} .

Output: time prediction \hat{t}_{n+1} , type prediction \hat{k}_{n+1}

▷ If we are predicting using MCMC,

$$G(\mathbf{x}, \Theta^G) = p(\mathbf{z} | \mathbf{x}; \Theta), \text{ where } \Theta^G = \Theta.$$

▷ If we are predicting using approximation,

$$G(\mathbf{x}, \Theta^G) = q(\mathbf{z}; \mathbf{x}, \Theta^I), \text{ where } \Theta^G = \Theta^I.$$

- 1: **for** $s = 1$ **to** \mathcal{S} **do**
 - 2: sample $\mathbf{z}^s \sim G(\mathbf{x}, \Theta^G)$
 - 3: **end for**
 - 4: estimate the CIFs for the top layer based on $\{\mathbf{z}^s\}_{s=1}^{\mathcal{S}}$ (MLE)
 - 5: **for** $s = 1$ **to** \mathcal{S} **do**
 - 6: sample $\mathbf{z}^s \sim G(\mathbf{x}, \Theta^G)$
 - 7: sample the future time \hat{t}_{n+1}^s based on \mathbf{z}^s
 - 8: sample the future type \hat{k}_{n+1}^s based on \mathbf{z}^s
 - 9: **end for**
 - 10: $\hat{t}_{n+1} = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} \hat{t}_{n+1}^s$
 - 11: $\hat{k}_{n+1} = \arg \max_{k \in \{1, \dots, K_0\}} \sum_{s=1}^{\mathcal{S}} \mathbb{1}_k(\hat{k}_{n+1}^s)$
-

Remark 1. Similar to Hong and Shelton (2022), we can simply replace the kernel function with a non-causal kernel (i.e., a kernel function that has positive values for all inputs with any dimensionality) to apply UNSPs in general Euclidean space (not just a timeline). It is not as straightforward to apply USAPs to spatial point processes (SPPs) with multiple dimensions, as we need to identify the “bounding” events for any point in space.

5 INFERENCE

5.1 Inference for the Model Parameters

While Hong and Shelton (2022) view the line 3 in Alg. 1 as a part of ascent-based MCEM, we can also view it as an unbiased estimator of the gradient of the marginal likelihood $\log p(\mathbf{x}; \Theta)$ based on the Fisher identity (Ou and Song, 2020; Naesseth et al., 2020),

$$\nabla_{\Theta} \log p(\mathbf{x}; \Theta) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x};\Theta)} [\nabla_{\Theta} \log p(\mathbf{z}, \mathbf{x}; \Theta)]. \quad (7)$$

Based on Eq. 7, we can update the model parameters by stochastic gradient ascent. The full convergence analysis for the model parameters has not been well-established, and it is still an active research area (Neath et al., 2013; Hong and Shelton, 2022).

5.2 Inference for the Variational Parameters

Variational inference with the inclusive KL divergence and unbiased gradient has demonstrated the ability to help mitigate the issue of the underestimation of the variance of the posterior (Naesseth et al., 2020). The inclusive KL divergence between the true posterior point processes and the approximate point processes $\text{KL}(p \parallel q^I)$ is

$$\mathbb{E}_{p(\mathbf{z}|\mathbf{x};\Theta)} [\log p(\mathbf{z} | \mathbf{x}; \Theta) - \log q(\mathbf{z}; \mathbf{x}, \Theta^I)],$$

where $\Theta^I = \left\{ \left\{ \Theta_{\ell,k}^I \right\}_{\ell=1}^L \right\}_{k=1}^{K_\ell}$, $\log q(\mathbf{z}; \mathbf{x}, \Theta^I)$ is the log-likelihood of the approximate point processes, and

$$\begin{aligned} \log q(\mathbf{z}; \mathbf{x}, \Theta^I) &= \sum_{\ell=1}^L \log q(\mathbf{z}_\ell; \mathbf{z}_{\ell-1}, \Theta_\ell^I) \\ &= \sum_{\ell=1}^L \sum_{k=1}^{K_\ell} \left(\sum_{t_\ell, k, j} \log \lambda_{\ell, k}^I(t_\ell, k, j) - \int_0^T \lambda_{\ell, k}^I(t) dt \right). \end{aligned}$$

The gradient of the KL divergence *w.r.t* the variational parameters Θ^I is

$$\mathbb{E}_{p(\mathbf{z}|\mathbf{x};\Theta)} [-\nabla_{\Theta^I} \log q(\mathbf{z}; \mathbf{x}, \Theta^I)], \quad (8)$$

because $p(\mathbf{z} | \mathbf{x}; \Theta)$ does not depend on Θ^I .

Naesseth et al. (2020) prove the convergence of the variational parameters under some regularity conditions for MCMC with a fixed number of dimensions. However, our Markov chains have an unbounded number of dimensions. We leave the research for the convergence of our variational parameters for future work.

According to Eq. 8 and line 4 in Alg. 1, we can see that the gradient of the inclusive KL divergence is identical to the gradient of the log-likelihood of the virtual point processes if we let the virtual point processes be our approximate point processes (i.e., let $\tilde{\Theta} = \Theta^I$). Therefore, we

can use Alg. 1 to train our approximate point processes and the parameters for the virtual point processes would just be the parameters for our approximate point processes. Then, when doing sampling for the approximate posterior point processes, we can directly sample from it using the inversion sampling without involving any gradient ascent or MCMC steps. This is especially helpful when doing prediction because we can avoid the time-consuming mixing steps of MCMC for the prediction of each event. We will explain more about this in Sec. 6.

We use Adam (Kingma and Ba, 2015) to optimize both the model parameters and variational parameters.

6 PREDICTION

We follow the prediction procedures with MCMC developed by Hong and Shelton (2022) in Alg. 2. Suppose we are given a sequence of events $\{e_1, e_2, \dots, e_n\}$, where $e_i = (t_i, k_i)$, t_i and k_i represent the time and the type for the i -th event respectively, and $t_1 \leq t_2 \leq t_3 \leq \dots \leq t_n$. We initialize RPPs with parameters Θ , VPPs with parameters $\tilde{\Theta}$. These parameters were obtained by MCEM running on the training data. We first generate posterior samples to estimate the constant rates for the top layer. Suppose the number of events for the s -th sample of $Z_{L,k}$ is $m_{L,k}^s$ and the time period is $[0, T]$, then the MLE for the constant rate is $\mu_k = \frac{1}{S} \sum_{s=1}^S \frac{m_{L,k}^s}{T}$. After getting new constant rates for the top layer, we can generate a new set of posterior samples, starting the initial state of the Markov chain to be the last sample of the previous MCMC sampling. Conditional on the generated posterior sample \mathbf{z} , we can extend the CIFs of RPPs to the future (*i.e.*, the CIFs at the time period (t_n, ∞)), because the CIFs only depend on the history of the events. Then, we can sample the time and the type for the next future event. Suppose the samples for the next future event (e_{n+1}) are $(t_{n+1}^1, k_{n+1}^1), (t_{n+1}^2, k_{n+1}^2), \dots, (t_{n+1}^S, k_{n+1}^S)$, where (t_{n+1}^s, k_{n+1}^s) represents the s -th sample of the time and the type for the $(n+1)$ -th event, then the time prediction is $\hat{t}_{n+1} = \frac{1}{S} \sum_{s=1}^S t_{n+1}^s$ and the type prediction is $\hat{k}_{n+1} = \arg \max_{k \in \{1, \dots, K_0\}} \sum_{s=1}^S \mathbb{1}_k(k_{n+1}^s)$, where $\mathbb{1}_k(k_{n+1}^s)$ is equal to 1 iff $k = k_{n+1}^s$. After each prediction, we record the last posterior sample as the initial state of the next MCMC step, and we also update the constant rates for the top layer using MLE.

In Alg. 2, we can replace the MCMC sampling with the sampling from our approximate posterior point processes to get our proposed method for prediction. Sampling from $q(\mathbf{z}; \mathbf{x}, \Theta^I)$ is much faster than sampling from MCMC, because we can directly use the inversion method (Çinlar, 2013) to sample instead of performing many MCMC steps. Faster sampling can help us achieve better prediction performance when only a limited amount of time is allowed,

which we will use experiments to demonstrate in Sec. 7.

7 EXPERIMENTS

The code is available online at https://github.com/hongchengkuan/Inclusive_VI_NSPPs. We implement all the algorithms on the same code base using Pytorch, so the implementation has little influence on the time comparison.

Following Hong and Shelton (2022), we use root mean square error (RMSE) to compare the time prediction and use accuracy to compare the type prediction. The models we use for DNSPs are 1-hidden and 2-hidden as explained in Hong and Shelton (2022). Each dataset is split into training, validation, and test sets. We use training sets to train our model parameters and variational parameters, use validation sets to stop early, and use test sets to calculate the metrics used to compare the performance. When performing prediction, we use different numbers of samples to measure the performance. Different numbers of samples correspond to different computational time budgets, as more time is needed for a larger sample size. We add a tiny background base rate (1×10^{-10}) to the intensity for the observation to prevent NaN and Inf issues. More details about the experiments can be found in Appx. B.

In Figs. 2, 3, 4 and 5, we use different line styles with different colors to represent the results from different models or algorithms, where DNSP-MCMC represents the results obtained by performing prediction using MCMC for DNSPs, DNSP-UNSP represents the results obtained by performing prediction using approximation with UNSPs for DNSPs, DNSP-USAP represents the results obtained by performing prediction using approximation with USAPs, THP represents transformer Hawkes process (Zuo et al., 2020), SAHP represents self-attentive Hawkes process (Zhang et al., 2020), NHP represents neural Hawkes process (Mei and Eisner, 2017), and MTPP is a multitask point process model (Lian et al., 2015). The vertical axes represent the RMSE or accuracy. The horizontal axes represent the time used to do sampling with various sample sizes for the predictions for all events from all sequences. Among UNSPs, USAPs, and MCMC, UNSPs are the best in the areas filled with light orange, and USAPs are the best in the areas filled with light green. A table of results with performance compared at fixed time-points is presented in Appx. B.3, where “/” means there is no result for that entry.

Our experiments show that when only a limited amount of time is available, UNSPs and USAPs perform better than MCMC for both time prediction and type prediction. USAPs are clearly better than UNSPs in terms of time prediction, and the type predictions of USAPs and UNSPs are similar, although USAPs require more computational resources than UNSPs. Moreover, DNSPs-based methods are

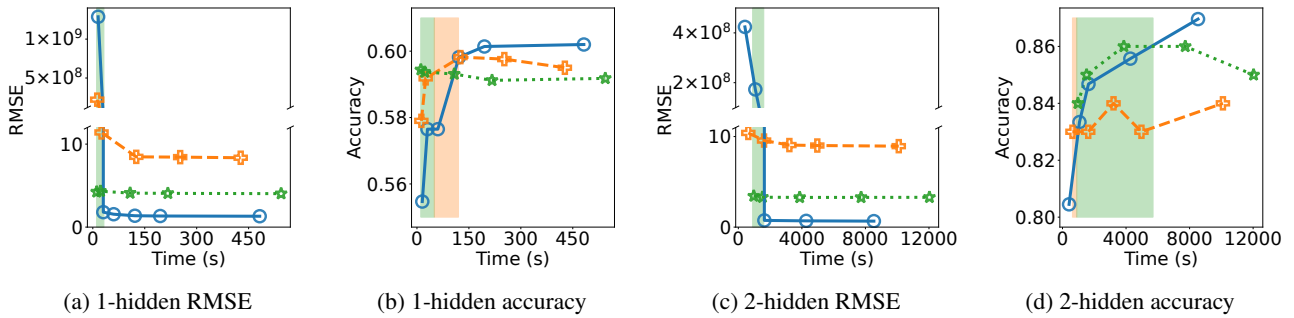


Figure 2: Synthetic Dataset

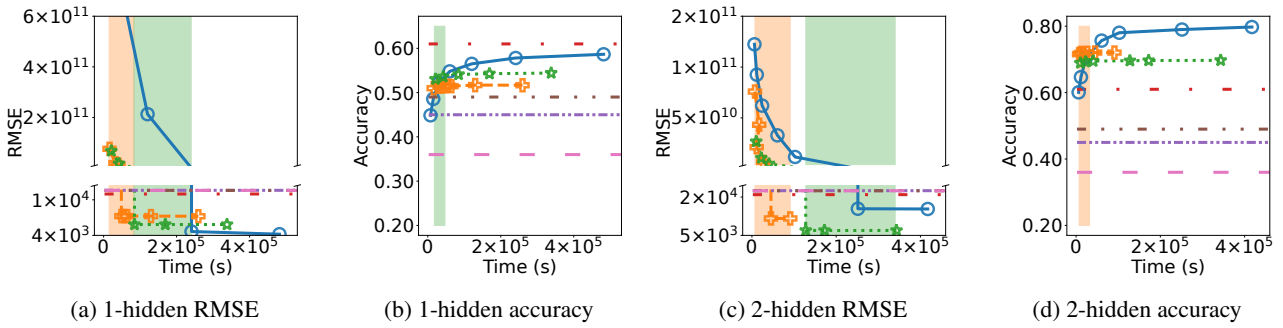


Figure 3: Retweet Dataset

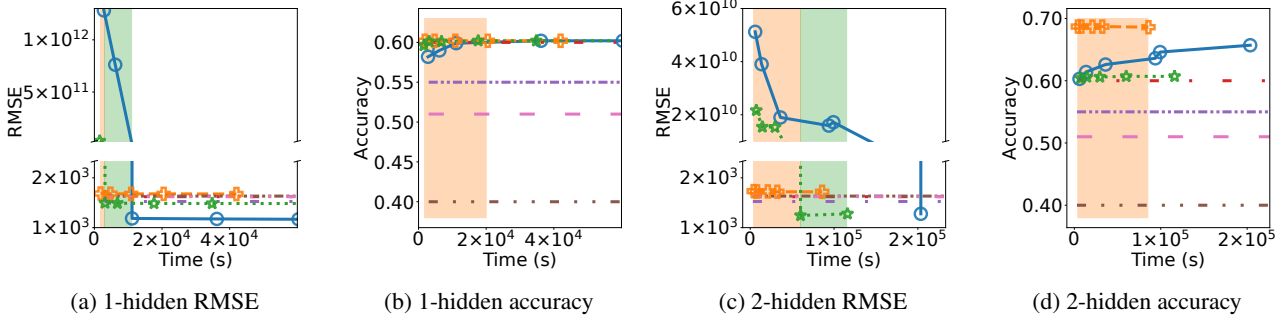


Figure 4: Earthquake Dataset

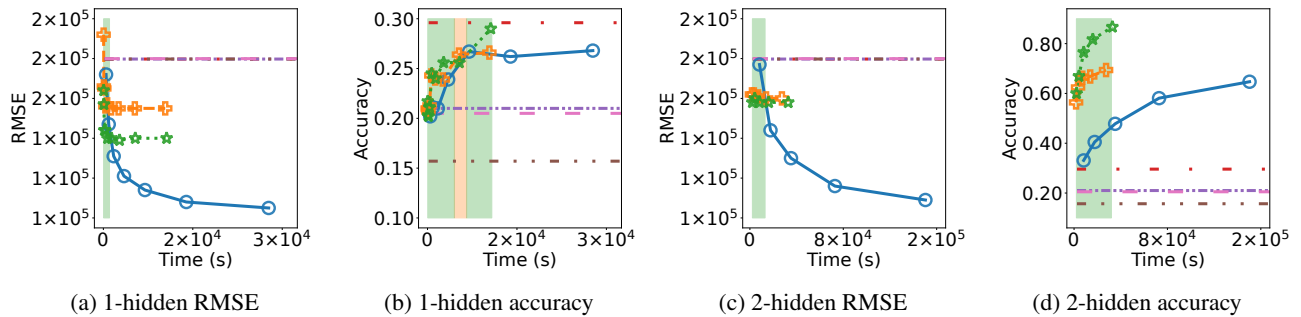


Figure 5: Homicide Dataset

better than non-DNSPs-based methods for all these real-world datasets.

7.1 Synthetic Data Experiments

We construct 2 synthetic datasets. One dataset is generated from a 1-hidden model and the other is generated from a 2-hidden model. We use the 1-hidden DNSP to learn from the dataset generated from 1-hidden, and the 2-hidden DNSP to learn from the dataset generated from 2-hidden. For simplicity, k is set to be 1 and fixed for the Weibull kernel, so that it becomes an exponential function.

Fig. 2 summarizes the results for our synthetic datasets. It shows that when we increase the sample size, the prediction performance of UNSPs, USAPs, and MCMC become better, at a cost of running time. The improvement of MCMC is more significant, as MCMC becomes closer to the true posterior point processes when we sample more from MCMC, while USAPs and UNSPs will never converge to the true posterior point processes. It also demonstrates that USAPs and UNSPs can achieve better prediction results for both the time and the type than MCMC when only a small period of time is allowed. When we can run our programs long enough, MCMC achieves the best results for all these experiments, which is reasonable because MCMC converges to the true posterior distribution when the sample size goes to infinity and the synthetic data is generated from DNSPs.

USAPs are better than UNSPs with regards to RMSE for both 1-hidden and 2-hidden datasets (Figs. 2a and 2c). There is a significant drop for the RMSE when we increase the sample size; it is because we may have no or very few hidden events in our posterior samples when the sample size is too small, causing the CIFs for the top layer or for the future to be very small. With a small CIF, the sample for the next future event would be very far away from the next true future event.

For accuracy, USAPs are clearly better than UNSPs for the 2-hidden synthetic dataset (Fig. 2d) and they have similar performance for the 1-hidden synthetic dataset (Fig. 2b).

7.2 Real-World Data Experiments

Similar to Hong and Shelton (2022), the datasets we use are of retweets (Zhao et al., 2015), earthquakes (NCEDC, 2014; BDSN, 2014; HRSN, 2014; BARD, 2014), and homicides (COC, 2022). More details of the datasets can be found in Appx. B. We compare MCMC, UNSPs, and USAPs for DNSPs with other state-of-the-art methods: transformer Hawkes process (THP) (Zuo et al., 2020), self-attentive Hawkes process (SAHP) (Zhang et al., 2020), neural Hawkes process (NHP) (Mei and Eisner, 2017), and MTPP (Lian et al., 2015). THP, SAHP, and NHP are neural-network-based Hawkes processes. MTPP is a Cox-

process-based model. We split each real-world dataset into training, validation, and test sets. We train, validate and test for all these models using the same split. For retweets, we use mini-batch gradient ascent, and we use batch gradient ascent for other datasets. The training and prediction procedures of the methods are the same as in Hong and Shelton (2022). We do not fix k for the Weibull kernel as we did in the synthetic data experiments.

Figs. 3, 4 and 5 summarize the experimental results for retweets, earthquakes, and homicides respectively. Similar to the experimental results for the synthetic dataset, UNSPs and USAPs are better than MCMC for both the time prediction and the type prediction when we only have a small number of samples from the approximate posterior distribution. As we increase the sample size, UNSPs, USAPs, and MCMC all gain some improvement to various degrees.

For the time prediction, DNSPs with MCMC or approximate posterior point processes achieve the best prediction results in all these experiments (Figs. 3a, 3c, 4a, 4c, 5a, and 5c). UNSPs or USAPs are better than other non-DNSPs baselines for all these experiments (Figs. 3a, 3c, 4a, 4c, 5a, and 5c). USAPs are better than UNSPs for all of these experiments in the end (Figs. 3a, 3c, 4a, 4c, 5a, and 5c). UNSPs are better than USAPs for retweets and earthquakes when we only have a small number of samples (Figs. 3a, 3c, 4a, and 4c). USAPs are always better than UNSPs for homicides in terms of time prediction (Figs. 5a and 5c).

For the type prediction, DNSPs with MCMC or approximate posterior point processes achieve the best prediction results for all these datasets (Fig. 3d for retweets, 4d for earthquakes, Fig. 5d for homicides). UNSPs or USAPs are better than other non-DNSPs baselines for all these datasets (Fig. 3d for retweets, 4d for earthquakes, Fig. 5d for homicides). In Figs. 3b and 5d, USAPs are better than UNSPs, while UNSPs are better than USAPs in Figs. 3d and 4d. USAPs and UNSPs have similar performance in Figs. 4b and 5b.

Compared with the results in Hong and Shelton (2022), DNSP-MCMC in this paper is better in terms of accuracy and RMSE. The performance of MCMC differs because we use a Weibull kernel in this paper, while Hong and Shelton (2022) adopt a gamma kernel, as explained in Sec. 2.3.

7.3 Computational Complexity

The full analysis requires the bound for the mixing steps of MCMC. The analysis is not trivial and we do not have this bound (see Hong and Shelton, 2022, Appx. G.5).

8 CONCLUSION

VI and point processes have both attracted great attention due to their excellent performance. However, little atten-

tion has been given to developing VI algorithms for point processes with hierarchical structures. We develop the first VI algorithm for NSPs (a class of point processes with hierarchical structures) in this paper.

Typically, posterior inference for NSPs is considered a very hard problem, and MCMC is required, as it involves an unbounded number of points in the posterior point processes. We incorporate MCMC into our VI algorithm, treating the samples from our approximate posterior point processes as the candidates for the posterior point processes. During training processes, we gradually update our approximations to make our approximations become closer and closer to the posterior point processes through the minimization of the inclusive KL divergence. Our experiments show that our approximate posterior point processes (USAPs and UNSPs) can fit the true posterior point processes very well. When time constraint is a concern, USAPs and UNSPs provide a very good alternative to MCMC.

In our VI algorithm, we bring 4 active research areas (MCMC, VI, neural networks, and point processes) together, and we have found many research topics that we believe will be of interest to many researchers in these areas, *e.g.*, analysis of the mixing time of MCMC, convergence analysis of variational parameters, efficient and well-behaved architectures of neural networks, and the construction of spatio-temporal point processes with hierarchies.

Acknowledgements

Computational resources were provided through Office of Naval Research grant N00014-18-1-2252.

Data for this study came from the Berkeley Digital Seismic Network (BDSN), doi:10.7932/BDSN; the High Resolution Seismic Network (HRSN), doi:10.7932/HRSN; and the Bay Area Regional Deformation Network (BARD), doi:10.7932/BARD, all operated by the UC Berkeley Seismological Laboratory and archived at the Northern California Earthquake Data center (NCEDC), doi:10.7932/NCEDC.

References

- Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.
- BARD (2014). Bay area regional deformation network. UC Berkeley Seismological Laboratory. dataset.
- BDSN (2014). Berkeley digital seismic network. UC Berkeley Seismological Laboratory. dataset.
- Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.
- Çinlar, E. (2013). *Introduction to stochastic processes*. Dover Publications.
- Chen, R. T. Q., Amos, B., and Nickel, M. (2021). Neural spatio-temporal point processes. In *International Conference on Learning Representations*.
- COC (2022). City of Chicago, Crimes - 2001 to present. <https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2>. Accessed: 2022-08-14.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (GELUs).
- Hong, C. and Shelton, C. (2022). Deep Neyman-Scott processes. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3627–3646. PMLR.
- HRSN (2014). High resolution seismic network. UC Berkeley Seismological Laboratory. dataset.
- Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.
- Lian, W., Henao, R., Rao, V., Lucas, J., and Carin, L. (2015). A multitask point process predictive model. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2030–2038, Lille, France. PMLR.
- Linderman, S. W., Wang, Y., and Blei, D. M. (2017). Bayesian inference for latent Hawkes processes. In *Advances in Approximate Bayesian Inference*.
- Mei, H. and Eisner, J. M. (2017). The neural Hawkes process: A neurally self-modulating multivariate point process. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Møller, J. and Waagepetersen, R. P. (2003). *Statistical inference and simulation for spatial point processes*. CRC Press.
- Naesseth, C., Lindsten, F., and Blei, D. (2020). Markovian score climbing: Variational inference with KL(p ||

- q). In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15499–15510. Curran Associates, Inc.
- NCEDC (2014). Northern California earthquake data center. UC Berkeley Seismological Laboratory. dataset.
- Neath, R. C. et al. (2013). On convergence properties of the Monte Carlo EM algorithm. In *Advances in modern statistical theory and applications: a Festschrift in Honor of Morris L. Eaton*, pages 43–62. Institute of Mathematical Statistics.
- Neyman, J. and Scott, E. L. (1958). Statistical approach to problems of cosmology. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(1):1–29.
- Ou, Z. and Song, Y. (2020). Joint stochastic approximation and its application to learning discrete latent variable models. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 929–938. PMLR.
- Park, J., Chang, W., and Choi, B. (2022). An interaction Neyman–Scott point process model for coronavirus disease-19. *Spatial Statistics*, 47:100561.
- Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In Kaski, S. and Corander, J., editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland. PMLR.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Williams, A., Degleris, A., Wang, Y., and Linderman, S. (2020). Point process models for sequence detection in high-dimensional neural spike trains. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14350–14361. Curran Associates, Inc.
- Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. (2020). Self-attentive Hawkes process. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11183–11193. PMLR.
- 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*, pages 1513–1522.
- Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer Hawkes process. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11692–11702. PMLR.

A NETWORK STRUCTURES FOR USAPS

Event Embedding Each event $e_{\ell,i,j} = (t_{\ell,i,j}, \text{ppid}_{\ell,i} = \sum_{w=0}^{\ell-1} K_w + i)$ consists of the time $t_{\ell,i,j}$ and the identifier $\text{ppid}_{\ell,i}$. Similar to Vaswani et al. (2017); Zuo et al. (2020); Zhang et al. (2020), we first encode the event time into a d_M -dimensional vector $\mathbf{pe}_{\ell,i,j}$ through positional encoding,

$$\mathbf{pe}_{\ell,i,j}^k = \begin{cases} \cos(t_{\ell,i,j}/10000^{\frac{k-1}{d_M}}), & \text{if } k \text{ is odd,} \\ \sin(t_{\ell,i,j}/10000^{\frac{k}{d_M}}), & \text{if } k \text{ is even,} \end{cases}$$

where $\mathbf{pe}_{\ell,i,j}^k$ is the k -th dimension of $\mathbf{pe}_{\ell,i,j}$.

We also train an embedding matrix $W_{eid} \in \mathbb{R}^{d_M \times n_{pp}}$, where $n_{pp} = \sum_{w=0}^L K_w$, for the identifiers. For each event with identifier $\text{ppid}_{\ell,i}$, the embedding for the identifier is $W_{eid} \cdot \mathbf{p}_{\ell,i}$, where $\mathbf{p}_{\ell,i} \in \mathbb{R}^{n_{pp} \times 1}$ is a one-hot vector. The $(\sum_{w=0}^{\ell-1} K_w + i)$ -th entry of $\mathbf{p}_{\ell,i}$ is 1, and the other entries are all 0.

To incorporate the information from both the time and the identifier, the embedding $\mathbf{x}_{\ell,i,j}^e$ of each event $e_{\ell,i,j}$ can be represented as

$$\mathbf{x}_{\ell,i,j}^e = \mathbf{pe}_{\ell,i,j} + W_{eid} \cdot \mathbf{p}_{\ell,i}.$$

Self-Attention The parameters of the intensity function $\lambda_{\ell,k}^I(t)$ are determined by all the events from the point processes which are connected to the point process $Z_{\ell-1,i}$. For each interval $(t_{\ell-1,i,j-1}, t_{\ell-1,i,j}]$, we use self-attention (Vaswani et al., 2017; Zhang et al., 2020; Zuo et al., 2020) to encode the events in the layer immediately below to a hidden vector,

$$\mathbf{h}_{\ell-1,i,j}^a = \left(\sum_{t=1}^{m_{\ell-1,i}} f(W_q \cdot \text{LayerNorm}(\mathbf{x}_{\ell-1,i,j}^e), W_k \cdot \mathbf{x}_{\ell-1,i,t}^e) \cdot (W_v \cdot \mathbf{x}_{\ell-1,i,t}^e) \right) / \sum_{t=1}^{m_{\ell-1,i}} f(W_q \cdot \text{LayerNorm}(\mathbf{x}_{\ell-1,i,j}^e), W_k \cdot \mathbf{x}_{\ell-1,i,t}^e), \quad (9)$$

where $\text{LayerNorm}(\cdot)$ is a layer normalization (Ba et al., 2016) operation, $W_q \in \mathbb{R}^{d_k \times d_M}$, $W_k \in \mathbb{R}^{d_k \times d_M}$, and $W_v \in \mathbb{R}^{d_v \times d_M}$ are all linear transformation matrices that transfer the event embedding vectors to queries, keys and values respectively, and $f(\mathbf{x}_1, \mathbf{x}_2) = \exp(\mathbf{x}_1^T \mathbf{x}_2)$ is a similarity function to capture the relationship between a query and a key. Notice that unlike the self-attention adopted in (Zhang et al., 2020; Zuo et al., 2020), we not only consider the influence of the events that happened before the query, but the events that happened after the query. Because for each hidden point process, the posterior distribution of the hidden events can be influenced by all the events from the point processes that are connected to this hidden point process (Hong and Shelton, 2022, Proposition 1).

Eq. 9 works as a single-head self-attention, we can also concatenate multiple single-head attentions together to build a multi-head attention (Vaswani et al., 2017). We can rewrite Eq. 9 as

$$\mathbf{h}_{\ell-1,i,j}^a = \text{Self-Attention}(\mathbf{x}_{\ell-1,i,j}^e, W_q, W_k, W_v),$$

then the multi-head attention version for $\mathbf{h}_{\ell-1,i,j}^a$ would be

$$\begin{aligned} \mathbf{h}_{\ell-1,i,j}^a &= \text{Multi-Head-Self-Attention}(\mathbf{x}_{\ell-1,i,j}^e) \\ &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) \cdot W_o, \end{aligned} \quad (10)$$

where $\text{head}_d = \text{Self-Attention}(\mathbf{x}_{\ell-1,i,j}^e, W_q^d, W_k^d, W_v^d)$, $W_q^d \in \mathbb{R}^{d_k \times d_M}$, $W_k^d \in \mathbb{R}^{d_k \times d_M}$, $W_v^d \in \mathbb{R}^{d_v \times d_M}$, and $W_o \in \mathbb{R}^{hd_v \times d_M}$.

Multi-head self-attention can also be stacked into a deep structure, but we do not have this deep structure in our experiment.

Position-Wise Feed-Forward Layer The hidden vector $\mathbf{h}_{\ell-1,i,j}^a$ and the residual $\mathbf{x}_{\ell-1,i,j}^e$ are then fed into a position-wise feed-forward layer with the number of neurons in the hidden layer as d_H , generating the final output for the hidden vector,

$$\mathbf{h}_{\ell-1,i,j} = (W_2^F (\text{GELU}(W_1^F (\text{LayerNorm}(\text{input})) + \mathbf{b}_1^F)) + \mathbf{b}_2^F) + \text{input}, \quad (11)$$

where $\text{input} = \mathbf{h}_{\ell-1,i,j}^a + \mathbf{x}_{\ell-1,i,j}^e$, $W_1^F \in \mathbb{R}^{d_H \times d_M}$, $\mathbf{b}_1^F \in \mathbb{R}^{d_H \times 1}$, $W_2^F \in \mathbb{R}^{d_M \times d_H}$, $\mathbf{b}_2^F \in \mathbb{R}^{d_M \times 1}$, and GELU is the Gaussian Error Linear Unit (Hendrycks and Gimpel, 2016).

Output for the Parameters We apply a linear transformation to the final hidden vector $h_{\ell-1,i,j}$ to get the kernel parameters,

$$\theta_{\ell,k}^I = W_{\ell,k}^\theta h_{\ell-1,i,j} + \mathbf{b}_{\ell,k}^\theta,$$

where $W_{\ell,k}^\theta \in \mathbb{R}^{n_\theta \times d_M}$, $\mathbf{b}_{\ell,k}^\theta \in \mathbb{R}^{n_\theta \times 1}$, and n_θ is the number of parameters for $\theta_{\ell,k}^I$.

B EXPERIMENT DETAILS

B.1 Datasets

Synthetic Datasets The point processes are defined in the interval $(0, 20]$. The constant rate for the top layer is set to be 0.15. We fix the kernel parameters and k is set to be 1 for the Weibull kernel functions. The sampling procedure is the same as the generative process, and we do this for both the 1-hidden and 2-hidden models.

Retweets (Zhao et al., 2015) The retweet dataset consists of a set of tweet streams. The original tweet of a tweet stream has time 0, and the times of the other tweets are the retweet times relative to the original tweet. We have 3 types of events (small, medium, and large) for different numbers of followers. The “small” group consists of users with fewer than 120 users, the “medium” group consists of users with greater than 120 users but fewer than 1363 users, and the “large” group consists of the rest of the users.

Earthquakes (NCEDC, 2014; BDSN, 2014; HRSN, 2014; BARD, 2014) The earthquake dataset contains the times and magnitudes of the earthquakes collected from northern California earthquake catalog. The times of the earthquakes are constrained to be from 01/01/2014 00:00:00 to 01/01/2020 00:00:00. The region is spanning between 34.5° and 43.2° latitude and between -126.00° and -117.76° longitude. We divide the earthquakes into 2 types. The earthquakes with a magnitude smaller than 1 are small earthquakes, and the others are big earthquakes.

Homicides (COC, 2022) The homicide dataset includes the information of homicides that happened in Chicago. We choose 5 contiguous regions in Chicago with the most homicides. The time for the homicides is from 01/01/2001 00:00:00 to 01/01/2020 00:00:00. 5 types of this dataset correspond to 5 different regions in Chicago. The terms of use can be found in https://www.chicago.gov/city/en/narr/foia/data_disclaimer.html.

The statistics of the datasets are shown in Table 1.

Table 1: Datasets Statistics

DATASETS	# OF TYPES	# OF PREDICTIONS	# OF SEQUENCES		
			TRAINING	VALIDATION	TEST
1-HIDDEN SYNTHETIC	2	1563	1000	100	100
2-HIDDEN SYNTHETIC	2	4664	1000	100	100
RETWEETS	3	216465	20000	2000	2000
EARTHQUAKES	2	25646	209	53	53
HOMICIDES	5	997	6	2	2

B.2 Self-Attention Training Details

For all our experiments, we use 1 layer of multi-head self-attention block with 4 heads. The number of dimensions can be found in Table 2, where d_k , d_v , d_M , and d_H appear in Eqs. 10 and 11, MPSS stands for model parameters step size, and SASS stands for self-attention step size.

Table 2: Self-Attention Dimensions

DATASETS	MODEL	d_k	d_v	d_M	d_H	MPSS	SASS
1-HIDDEN SYNTHETIC	USAP(1-HIDDEN)	8	8	32	64	0.01	0.010
2-HIDDEN SYNTHETIC	USAP(2-HIDDEN)	8	8	32	64	0.01	0.010
RETWEETS	USAP(1-HIDDEN)	16	16	64	128	1.00	0.001
	USAP(2-HIDDEN)	16	16	64	128	1.00	0.001
EARTHQUAKES	USAP(1-HIDDEN)	16	16	64	128	1.00	0.001
	USAP(2-HIDDEN)	16	16	64	128	1.00	0.010
HOMICIDES	USAP(1-HIDDEN)	8	8	32	64	1.00	0.010
	USAP(2-HIDDEN)	16	16	64	128	1.00	0.010

B.3 Experiments

Table 3: 1-Hidden RMSE For Synthetic Dataset

MODEL	TIME(S)				
	16	32	64	128	256
DNSP-MCMC	1.24E+09	1.79E+00	1.54E+00	1.37E+00	1.33E+00
DNSP-UNSP	1.63E+08	1.12E+01	1.03E+01	8.46E+00	8.43E+00
DNSP-USAP	4.28E+00	4.30E+00	4.21E+00	4.09E+00	4.06E+00

Table 4: 1-Hidden Accuracy For Synthetic Dataset

MODEL	TIME(S)				
	16	32	64	128	256
DNSP-MCMC	5.56E-01	5.77E-01	5.78E-01	5.98E-01	6.02E-01
DNSP-UNSP	5.82E-01	5.92E-01	5.94E-01	5.98E-01	5.98E-01
DNSP-USAP	5.94E-01	5.94E-01	5.93E-01	5.93E-01	5.91E-01

Table 5: 2-Hidden RMSE For Synthetic Dataset

MODEL	TIME(S)				
	500	900	2000	4000	8000
DNSP-MCMC	3.91E+08	2.38E+08	7.33E-01	6.95E-01	6.64E-01
DNSP-UNSP	\	1.02E+01	9.44E+00	9.04E+00	8.97E+00
DNSP-USAP	3.44E+00	3.43E+00	3.30E+00	3.28E+00	3.28E+00

Table 6: 2-Hidden Accuracy For Synthetic Dataset

MODEL	TIME(S)				
	500	900	2000	4000	8000
DNSP-MCMC	8.08E-01	8.26E-01	8.48E-01	8.55E-01	8.68E-01
DNSP-UNSP	\	8.30E-01	8.32E-01	8.36E-01	8.36E-01
DNSP-USAP	8.40E-01	8.40E-01	8.52E-01	8.60E-01	8.59E-01

Table 7: Results Of Baselines For Retweet Dataset

MODEL	RMSE	ACCURACY
THP	1.56E+04	6.1E-01
SAHP	1.67E+04	4.5E-01
NHP	1.66E+04	4.9E-01
MTPP	1.66E+04	3.6E-01

Table 8: 1-Hidden RMSE For Retweet Dataset

MODEL	TIME(S)				
	21500	43000	86000	172000	250000
DNBP-MCMC	1.45E+12	9.07E+11	4.52E+11	1.22E+11	5.05E+03
DNBP-UNBP	5.80E+10	9.65E+09	9.42E+03	9.40E+03	9.39E+03
DNBP-USAP	6.72E+10	1.76E+10	7.05E+03	7.05E+03	7.04E+03

Table 9: 1-Hidden Accuracy For Retweet Dataset

MODEL	TIME(S)				
	21500	43000	86000	172000	250000
DNBP-MCMC	5.01E-01	5.33E-01	5.55E-01	5.71E-01	5.79E-01
DNBP-UNBP	5.11E-01	5.14E-01	5.15E-01	5.17E-01	5.17E-01
DNBP-USAP	5.30E-01	5.36E-01	5.41E-01	5.43E-01	5.43E-01

Table 10: 2-Hidden RMSE For Retweet Dataset

MODEL	TIME(S)				
	10300	20600	41200	82400	200000
DNBP-MCMC	9.93E+10	7.08E+10	4.84E+10	2.17E+10	3.98E+09
DNBP-UNBP	6.58E+10	1.24E+10	1.44E+09	9.42E+03	\
DNBP-USAP	2.66E+10	1.29E+10	4.76E+09	2.49E+09	6.28E+03

Table 11: 2-Hidden Accuracy For Retweet Dataset

MODEL	TIME(S)				
	10300	20600	41200	82400	200000
DNBP-MCMC	6.36E-01	6.90E-01	7.30E-01	7.69E-01	7.87E-01
DNBP-UNBP	7.16E-01	7.20E-01	7.20E-01	7.21E-01	\
DNBP-USAP	6.89E-01	6.92E-01	6.95E-01	6.95E-01	6.97E-01

Table 12: Results Of Baselines For Earthquake Dataset

MODEL	RMSE	ACCURACY
THP	1.93E+03	6.0E-01
SAHP	1.79E+03	5.5E-01
NHP	1.95E+03	4.0E-01
MTPP	1.93E+03	5.1E-01

Table 13: 1-Hidden RMSE For Earthquake Dataset

MODEL	TIME(S)				
	2900	6200	11100	36200	60000
DNBP-MCMC	1.27E+12	7.56E+11	5.10E+09	1.26E+03	1.25E+03
DNBP-UNBP	2.03E+03	2.02E+03	2.02E+03	2.01E+03	\
DNBP-USAP	4.83E+09	1.74E+03	1.73E+03	1.73E+03	1.73E+03

Table 14: 1-Hidden Accuracy For Earthquake Dataset

MODEL	TIME(S)				
	2900	6200	11100	36200	60000
DNBP-MCMC	5.82E-01	5.90E-01	5.99E-01	6.02E-01	6.02E-01
DNBP-UNBP	6.02E-01	6.02E-01	6.02E-01	6.02E-01	\
DNBP-USAP	6.00E-01	6.01E-01	6.01E-01	6.02E-01	6.02E-01

Table 15: 2-Hidden RMSE For Earthquake Dataset

MODEL	TIME(S)				
	7200	14400	28800	57600	116016
DNBP-MCMC	4.92E+10	3.83E+10	2.55E+10	1.78E+10	1.44E+10
DNBP-UNBP	2.09E+03	2.09E+03	2.08E+03	2.08E+03	\
DNBP-USAP	2.16E+10	1.54E+10	1.53E+10	1.21E+09	1.42E+03

Table 16: 2-Hidden Accuracy For Earthquake Dataset

MODEL	TIME(S)				
	7200	14400	28800	57600	116016
DNBP-MCMC	6.05E-01	6.14E-01	6.22E-01	6.30E-01	6.48E-01
DNBP-UNBP	6.87E-01	6.87E-01	6.87E-01	6.87E-01	\
DNBP-USAP	6.04E-01	6.06E-01	6.06E-01	6.07E-01	6.07E-01

Table 17: Results Of Baselines For Homicide Dataset

MODEL	RMSE	ACCURACY
THP	1.80E+05	2.96E-01
SAHP	1.80E+05	2.10E-01
NHP	1.80E+05	1.57E-01
MTPP	1.80E+05	2.05E-01

Table 18: 1-Hidden RMSE For Homicide Dataset

MODEL	TIME(S)				
	483	966	1932	3864	5800
DNSP-MCMC	1.72E+05	1.47E+05	1.31E+05	1.21E+05	1.17E+05
DNSP-UNSP	1.56E+05	1.56E+05	1.55E+05	1.55E+05	1.55E+05
DNSP-USAP	1.43E+05	1.40E+05	1.40E+05	1.39E+05	1.40E+05

Table 19: 1-Hidden Accuracy For Homicide Dataset

MODEL	TIME(S)				
	483	966	1932	3864	5800
DNSP-MCMC	2.02E-01	2.11E-01	2.11E-01	2.40E-01	2.55E-01
DNSP-UNSP	2.25E-01	2.42E-01	2.40E-01	2.49E-01	2.64E-01
DNSP-USAP	2.21E-01	2.44E-01	2.45E-01	2.56E-01	2.57E-01

Table 20: 2-Hidden RMSE For Homicide Dataset

MODEL	TIME(S)		
	8300	16000	27000
DNSP-MCMC	1.77E+05	1.50E+05	1.37E+05
DNSP-UNSP	1.61E+05	1.60E+05	1.60E+05
DNSP-USAP	1.58E+05	1.58E+05	1.58E+05

Table 21: 2-Hidden Accuracy For Homicide Dataset

MODEL	TIME(S)		
	8300	16000	27000
DNSP-MCMC	3.32E-01	3.92E-01	4.44E-01
DNSP-UNSP	6.64E-01	6.72E-01	6.93E-01
DNSP-USAP	7.65E-01	8.17E-01	8.50E-01

B.4 Hardware and Software

We run the experiments for synthetic datasets, retweets, and earthquakes in a cluster. For each job, we use two cores from a Intel® Xeon® Silver 4214 CPUs running at 2.20GHz and 1 GeForce® RTX 2080 Ti.

We run each experiment for homicides in a machine with a core from Intel® i7-5930K CPU and 1 GeForce® GTX 1080 Ti.

We use Pytorch to implement our algorithms.

B.5 Implementation of MCMC for GPUs

We re-implemented the MCMC algorithm to use GPUs. We do not randomly select a move from resampling, flip, and swap, instead, we have a deterministic order for these moves. We choose this setting to minimize the number of dimension changes of the tensors used to store the events.

We store the real events and the virtual events in the same tensor. The only move that changes the dimension of this tensor is to re-sample the virtual events. For each MCMC sampling step, we first do a re-sampling, then followed by 3 flips, 1 swap, 3 flips, and 1 swap.

C SOCIETAL IMPACT

Our proposed inference algorithm can be used to predict future earthquakes, violent crimes, and severe medical complications when speed is the top concern. The successful application of our method can help save a lot of lives.

The misuse of our algorithm may also cause some problems. For example, some companies may use our algorithm to predict the customers' behavior and manipulate their buying activity, so customers may tend to buy more than what they need and then can cause some waste.