UNIVERSITY OF CALIFORNIA
RIVERSIDE

Deep Neyman-Scott Processes

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Chengkuan Hong

September 2022

Dissertation Committee:

Dr. Christian R. Shelton, Chairperson
Dr. Evangelos Papalexakis
Dr. Yehua Li
Dr. Tao Jiang

The Dissertation of Chengkuan Hong is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, Dr. Christian R. Shelton, who guided me through the maze of point processes. He introduced this interesting and profound topic to me and accompanied me to accumulate every piece of the building block of this thesis. Without his unwavering encouragement and patience, I would never have had such an amazing research experience.

I would also like to thank my other dissertation committee members: Dr. Evangelos Papalexakis, Dr. Yehua Li, and Dr. Tao Jiang. They raised many interesting questions in my defense for the proposal and the final dissertation and gave me great encouragement. I also had nice discussions with Dr. Papalexakis and Dr. Li about the research topics and career plans, from which I learned a lot.

I am also thankful to the other members of R-LAIR: Dave Gomboc, Jing Jin, Marjuka Lazin, Harini Venkatesan, Mike Izbicki, Leah Fauber, Mehran Ghamaty, Gaurav Jhaveri, Colin Lee, Sanjana Sandeep, Chandini Shetty, and Anthony Williams. My lab mates are great companions and they gave me a lot of valuable feedback.

Last but not least, I could not have undertaken this journey without my mother, Lianfang Shan, and my father, Zhihao Hong. They support me both financially and mentally. They always stand by me no matter what happened. There would never be enough space to express my gratitude to them.

The text of this dissertation, in part, is a reprint of the material as it appears in “Deep Neyman-Scott Processes” (published on *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, PMLR 151:3627-3646, 2022.*, May 3,

2022). The co-author Christian R. Shelton listed in that publication directed and supervised the research which forms the basis for this dissertation.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Deep Neyman-Scott Processes

by

Chengkuan Hong

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2022
Dr. Christian R. Shelton, Chairperson

Building hierarchical models has long been of interest to members of the artificial intelligence community. Many hierarchical models (*e.g.*, probabilistic graphical models, deep neural networks, and graph neural networks) have achieved great success. However, all of the current hierarchical models assume the number of variables is fixed. The natural question is "What if the number of variables is unknown *a priori*?"

For this purpose, we consider a *deep Neyman-Scott process* in this thesis, for which the building components of a network are all Poisson processes. The number of variables is a random variable.

We develop an efficient posterior sampling via Markov chain Monte Carlo and use it for likelihood-based inference. Our method allows for the inference in sophisticated hierarchical point processes. We show in the experiments that more hidden Poisson processes yield better performance for likelihood fitting and event types prediction. We also compare our method with recently developed neural-network-based models for temporal real-world datasets and demonstrate competitive abilities for both data fitting and prediction.

We also show that we can use our likelihood-based inference algorithm to learn approximate posterior point processes. Our approximate posterior point processes are close to the true posterior point processes. When doing prediction, we no longer have to do Markov chain Monte Carlo to approximate the true posterior point processes. Instead, we can directly sample our approximate posterior point processes based on the observed data, and the prediction performance of our approximate posterior point processes is better than the prediction performance of the samples from Markov chain Monte Carlo when only a limited amount of running time is allowed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Point Processes

Point processes have attracted attention due to their ability to model the temporal and spatial patterns of data. They have been applied to various fields, *e.g.*, finance (Bauwens and Hautsch, 2009), neuroscience (Perkel et al., 1967), and cosmology (Stoica et al., 2014).

1.1 Temporal Point Processes

A *temporal point process* (TPP) (Daley and Vere-Jones, 2003) is usually represented as a counting process $N(t)$, *i.e.*

$$N(t) = \sum_{t_i \in \mathcal{H}_t} u(t - t_i),$$

where $\mathcal{H}_t = \{t_i \mid t_i < t\}$ is the history of events, $t_i \in \mathbb{R}_{\geq 0}$ is the time point where an event with index i happens, and

$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t \leq 0 \end{cases} .$$

We restrict our attention to the case when $N(t) < \infty$ and $t \in \mathbb{R}_{\geq 0}$ is finite.

Definition 1.1 (conditional intensity function)**.** *The* conditional intensity function *(CIF) is defined as*

$$\lambda(t) = \lim_{dt \to 0} \frac{\mathbb{E}(N([t, t+dt]) \mid \mathcal{H}_t)}{dt},$$

where $N([t, t+dt]) = N(t+dt) - N(t)$.

A CIF determines the expected number of events at each infinitesimal interval. We assume there can exist at most one event when $dt \to 0$ and the number of events follows a Bernoulli distribution. The probability of one event at $[t, t+dt]$ is $\lambda(t) \cdot dt$. Thus, a CIF fully determines a TPP.

Definition 1.2 (cumulative intensity function)**.** *The cumulative intensity function $\Lambda(t)$ of $\lambda(\cdot)$ is defined as*

$$\Lambda(t) = \int_0^t \lambda(\tau) d\tau.$$

1.2 Spatial Point Processes

In a similar fashion, we can define a *spatial point process* (SPP) as a random countable subset of a space S. Usually, S is a subset of a Euclidean space or a manifold. For a realization $\mathbf{x}$ of a SPP, $N(B)$ represents the cardinality of the points in $B \cap \mathbf{x}$. Here, we require that the realization to be *locally finite*, that is $N(B)) < \infty$, where $B \subseteq S$ is an arbitrary bounded region.

Definition 1.3 (Papangelou conditional intensity function (Papangelou, 1974))**.** *The* Papangelou conditional intensity function (PCIF) *of a SPP $\mathcal{N}$ at location ξ defined on Euclidean* *space is defined as*

$$\lambda_{\mathcal{P}}(\xi) = \lim_{\delta \to 0} \frac{\mathbb{E}(N(B_\delta(\xi)) \mid [\mathcal{N} \backslash B_\delta(\xi)])}{\nu(B_\delta(\xi))},$$

where $B_\delta(\xi)$ is the intersection of $\mathbf{x}$ and a ball $B_\delta(\xi)$ centered at ξ with radius δ, $[\mathcal{N} \backslash B_\delta(\xi)]$ is the information of the point processes $\mathcal{N}$ outside $B_\delta(\xi)$, and $\nu(B_\delta(\xi))$ is the volume of $B_\delta(\xi)$.

Similar to a CIF, a PCIF also uniquely determines an SPP. Different than CIF, PCIF considers the correlations of a point with other points in the whole space rather than only in the past.

1.3 Marked Point Processes

Let X be an SPP or a TPP on S. For a given space M, there is a random mark $m_\xi \in M$ attached to each point $\xi \in \mathbf{x} \sim X$, then

$$\mathcal{Y} = \{(\xi, m_\xi) : \xi \in \mathbf{x} \sim X\}$$

is called a marked point process.

1.4 Poisson Processes

A Poisson process is a special case of a point process such that the expected number of points at each infinitesimal interval is independent of all the other points, and the CIF is equal to the PCIF. Thus, we can just use an intensity function (IF) to describe a Poisson process, which is equal to the CIF and the PCIF.

If the IF $\lambda(t) = \lambda_0 > 0$ is a constant function, we call it a *homogeneous Poisson process* (HPP). If, otherwise, the $\lambda(t)$ varies with location or time, we call it an *inhomogeneous Poisson process* (IPP).

1.4.1 Density

For a Poisson process, there is no density *w.r.t* the Lebesgue measure as the realizations reside in infinite dimensional spaces. However, we can define densities through Radon–Nikodym derivative with the reference as another Poisson process.

Proposition 1.4 (Møller and Waagepetersen (2003))**.** *Let* $\Xi_i \sim \text{Poisson}(S, \lambda_i)$*,* $\lambda_i : S \to [0, \infty)$*, and* $\Lambda_i(S) = \int_S \lambda_i(\xi) d\xi < \infty$ *for* $i = 1, 2$*. Also* $\lambda_2(\xi) > 0$ *whenever* $\lambda_1(\xi) > 0$*. Then* $\text{Poisson}(S, \lambda_1)$ *is absolutely continuous w.r.t* $\text{Poisson}(S, \lambda_2)$*, with* density

$$f(\mathbf{x}) = \exp\left(\Lambda_2(S) - \Lambda_1(S)\right) \prod_{\xi \in \mathbf{x}} \lambda_1(\xi)/\lambda_2(\xi).$$

We usually choose $\lambda_2 = 1$ *s.t.* any Poisson process $\text{Poisson}(S, \lambda_1)$ in a bounded region S is absolutely continuous *w.r.t* $\text{Poisson}(S, 1)$ and the density is

$$f(\mathbf{x}) = \exp(\int_S d\xi - \Lambda_1(S)) \prod_{\xi \in \mathbf{x}} \lambda_1(\xi).$$

Since $\int_S d\xi$ is a constant number, we will write the density as

$$f(\mathbf{x}) = \exp(-\Lambda_1(S)) \prod_{\xi \in \mathbf{x}} \lambda_1(\xi) \tag{1.1}$$

without special specification.

1.4.2 Sampling

As we mainly focus on the TPPs, we only introduce the sampling for Poisson processes in one dimension. The sampling for an HPP is pretty straightforward. Suppose we want to generate samples for an HPP with intensity function as λ_0 on an interval $[0, T]$. We have two ways to sample:

- *Inter-event times.* The inter-event times follow an exponential distribution. So we only need to generate samples from the exponential distribution with mean $1/\lambda_0$.

- *Order statistics.* We can first generate the number n from a Poisson distribution with mean $\lambda_0 T$. Then we can generate n *i.i.d* random variables for the exact times from the uniform distribution on $[0, T]$.

Typical methods for the sampling for an IPP include the inversion method, thinning, and using order statistics:

- *Inversion method.* We can generate samples $z_1, z_2, \cdots$ from an HPP with 1 as its intensity function. Then $\Lambda^{-1}(z_1), \Lambda^{-1}(z_2), \cdots$ are the samples corresponding to the cumulative intensity function $\Lambda(\cdot)$.

- *Thinning.* Suppose we want to generate samples for a Poisson process with an intensity function as $\lambda(t)$. We first generate samples from a dominating Poisson process with an intensity function $\lambda_d(t)$ that is larger than the intensity function $\lambda(t)$ at any time point. Then we accept each sample at time t from the dominating Poisson process independently with probability $\frac{\lambda(t)}{\lambda_d(t)}$. This method requires that the dominating Poisson process should be sampled efficiently.

- *Order statistics.* For a given interval $[0, T]$, we first generate n from a Poisson distribution with expected value as $\Lambda(T)$. Then we generate n random variables from a distribution with cumulative distribution function $F(t) = \frac{\Lambda(t)}{\Lambda(T)}$ as order statistics.

1.5 Cox Processes

A Cox process (Cox, 1955), also called a doubly stochastic Poisson process, generalizes a Poisson process *s.t.* the intensity function of a Poisson process becomes a realization of a random field.

Definition 1.5 (Møller and Waagepetersen (2003)). *Given a nonnegative random field* $Z = \{Z(\xi) : \xi \in S \subseteq \mathbb{R}^d\}$*, if the conditional distribution of a point process* Ξ *is a Poisson process with intensity function* Z*, then* Ξ *is a* Cox process driven by the random field (or, latent process) Z.

A Cox process is generally preferred over a Poisson process when modeling real-world data, as we usually do not have any prior knowledge of the functional form of the intensity function for a Poisson process. A Cox process assumes the intensity function itself is a latent stochastic process. By introducing this latent process, we introduce dependence between the expected number of events at different times or places. Typical examples for the latent stochastic process are Poisson processes (Neyman and Scott, 1958), Cox processes (Adams, 2009), Strauss processes (Yau and Loh, 2012), and Gaussian processes (Adams et al., 2009).

Chapter 2

Deep Neyman-Scott Processes

The Neyman-Scott process (NSP) (Neyman and Scott, 1958) is a special class of Cox processes. In this thesis, we focus on Neyman-Scott processes (NSPs) with order larger than one, which we call deep Neyman-Scott processes (DNSPs). Neyman and Scott first described DNSPs in 1958; they were trying to model the distribution of galaxies in the Universe. Each building block of the DNSPs is a Poisson process. Taking DNSPs of order two as an example, one Poisson process generates the centers of some superclusters, consisting of galaxy clusters. In turn, each galaxy cluster generates its own set of points, galaxies, from a Poisson process centered the cluster center. The advantage of the DNSPs compared to the other Cox processes is that they have the ability to build deep hierarchical models where each component in the system is a point process.

2.1 The General Structure

To build DNSPs, we will stack Poisson processes in a hierarchical manner *s.t.* the distributions of the IFs as random processes are controlled by the Poisson processes on upper layers.

Each observed data point $\mathbf{x}$ is a collection of sequences $\{\mathbf{x}_k\}_{k=1}^{K_0} = \{\{t_{0,k,j}\}_{j=1}^{m_{0,k}}\}_{k=1}^{K_0}$, where $\mathbf{x}_k$ is the sequence of the k-th type of event (or events with the k-th mark in a discrete-marked point process) and $t_{0,k,j}$ is the time or the location of the j-th event of this type. (The 0 indicates this is an observation event at the bottom layer of the model.) For each data point $\mathbf{x}$, there are L hidden layers of point processes $\mathbf{Z} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_L\}$, with $\mathbf{Z}_\ell = \{Z_{\ell,k}\}_{k=1}^{K_\ell}$. K_ℓ is the number of hidden processes at level ℓ, and $Z_{\ell,k}$ is a Poisson process. $Z_{\ell,k}$ and $Z_{\ell+1,j}$ are connected by a kernel function $\phi_{\boldsymbol{\theta}_{(\ell+1,k)\to(\ell,j)}}(\cdot)$, whose functional form will be given later. Figure 2.1 gives the general structure of DNSPs similar to a graphical model, where each node represents a Poisson process.

2.2 Generative Model Semantics

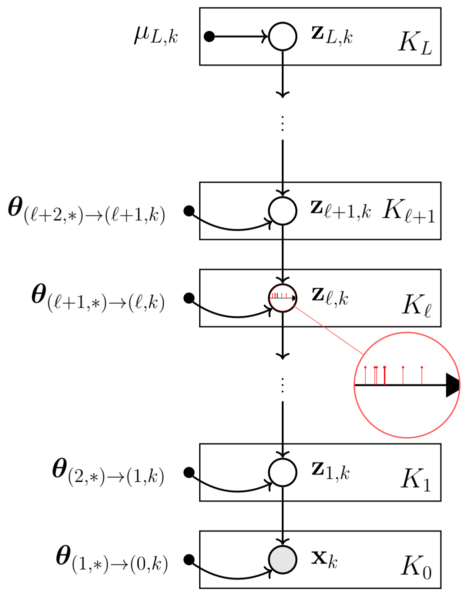
We first draw samples from the point processes on the top layer. The IF for $Z_{L,k}$ is a constant function

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

Next, we draw samples for each hidden Poisson process conditional on the Poisson process in the layer immediately above. The IF for $Z_{\ell,k}$ conditional on $\mathbf{Z}_{\ell+1}$ is

$$\lambda_{\ell,k}(t) = \sum_{i=1}^{K_{\ell+1}} \sum_{t_{\ell+1,i,j}} \phi_{\boldsymbol{\theta}_{(\ell+1,i)\to(\ell,k)}}(t - t_{\ell+1,i,j}), \tag{2.1}$$

where $\{t_{\ell+1,i,j}\}_{j=1}^{m_{\ell+1,i}}$ is a realization for $Z_{\ell+1,i}$. The filtration $\{\mathcal{H}_t\}$ for layer ℓ is $\{t_{\ell+1,i,j} : t_{\ell+1,i,j} < t\}$, *i.e.*, all events at the immediately upper layer at a prior time.



$\mu_{L,k}$
$\mathbf{z}_{L,k}$
K_L
$\boldsymbol{\theta}_{(\ell+2,*)\to(\ell+1,k)}$
$\mathbf{z}_{\ell+1,k}$
$K_{\ell+1}$
$\boldsymbol{\theta}_{(\ell+1,*)\to(\ell,k)}$
$\mathbf{z}_{\ell,k}$
K_ℓ
$\boldsymbol{\theta}_{(2,*)\to(1,k)}$
$\mathbf{z}_{1,k}$
K_1
$\boldsymbol{\theta}_{(1,*)\to(0,k)}$
$\mathbf{x}_k$
K_0

Figure 2.1: The structure of a DNSP, where $\boldsymbol{\theta}_{(\ell+1,*)\to(\ell,k)} = [\boldsymbol{\theta}_{(\ell+1,i)\to(\ell,k)}]_{i=1}^{K_{\ell+1}}$. The node for $\mathbf{z}_{\ell,k}$ is zoomed in.

Conditioned on $\mathbf{Z}_{\ell+1}$, $Z_{\ell,k}$ is a Poisson process whose IF at each time interval is independent of the IF at the other time intervals, and the number of events at each time interval follows a Poisson distribution.

The bounded region S for each hidden Poisson process is set to be the same as the evidence and no edge effects are considered in our model.

2.3 Evidence Likelihood

The observed data is assumed to be drawn conditioned on the hidden Poisson processes on the lowest hidden layer in a manner analogous to those of the layers above. The IF for $\mathbf{x}_k$ is

$$\lambda_{0,k}(t) = \sum_{i=1}^{K_1} \sum_{t_{1,i,j}} \phi_{\boldsymbol{\theta}_{(1,i)\to(0,k)}}(t - t_{1,i,j}), \tag{2.2}$$

where $\{t_{1,i,j}\}_{j=1}^{m_{1,i}}$ is a realization for $Z_{1,i}$.

Thus, according to Equation 1.1, the conditional log-likelihood for the data point $\mathbf{x}$ in the bounded region S is

$$\sum_{i=1}^{K_0} \left(\sum_{t_{0,i,j}} \log \lambda_{0,i}(t_{0,i,j}) - \int_S \lambda_{0,i}(t)dt \right).$$

2.4 Examples

Different kernels lead to different behaviors of DNSPs. Here we give different examples for kernels.

Example 2.1 (Thomas processes (Thomas, 1949))**.** *The kernel function is*

$$\phi_{\boldsymbol{\theta}} = \exp\left(-\|x\|^2/(2w^2)\right) / \left(2\pi w^2\right)^{d/2}, \textit{ for } w, d > 0,$$

where $\boldsymbol{\theta} = \{w, d\}$.

Example 2.2 (Matérn processes (Matérn, 1960))**.** *The kernel function is*

$$\phi_{\boldsymbol{\theta}}(x) = \frac{\mathbb{1}_{\|x\|\le r}}{w_d r^d}, \textit{ for } r, d > 0,$$

where $\boldsymbol{\theta} = \{r, d\}$.

Remark 2.3. *When we deal with the temporal data, it is reasonable to make the kernel function be zero for negative inputs (i.e., to be "causal") and converge to* 0 *as* t *goes to infinity, as we assume the influence of the events can only be from the past to the future and it will eventually disappear. The only constrain for the kernel function is that the sum appearing in the intensity function (Equations 2.1 and 2.2) should always be non-negative. We also want the kernel function to be as flexible as possible with only a few parameters.*

For the above given reasons, we give an example for TPPs in Example 2.4. The gamma kernel asymptotes to 0 as $x \to \infty$. Moreover, with the varying parameters, the gamma kernel can have either monotonically decreasing behavior, or a unimodal shape, which provides flexibility.

Example 2.4 (DNSPs with a gamma kernel for TPPs)**.** *The kernel function is*

$$\phi_{\boldsymbol{\theta}}(x) = \begin{cases} p \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \textit{ for } x > 0,\ p, \alpha, \beta > 0, \\ \\ 0,\ \textit{for } x \leq 0, \end{cases}$$

where $\boldsymbol{\theta} = \{p, \alpha, \beta\}$ *and* $\Gamma(\alpha)$ *is the gamma function.*

Figure 2.2 demonstrates the distributions of the events for DNSPs with two hidden layers. The forward sampling is implemented by using the inversion method (Çinlar, 2013). Please refer to Section 4.1 for how to do posterior sampling.

Example 2.5 (DNSPs with a Weibull kernel for TPPs)**.** *The kernel function is*

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

where $\boldsymbol{\theta} = \{p, k, \lambda\}$.

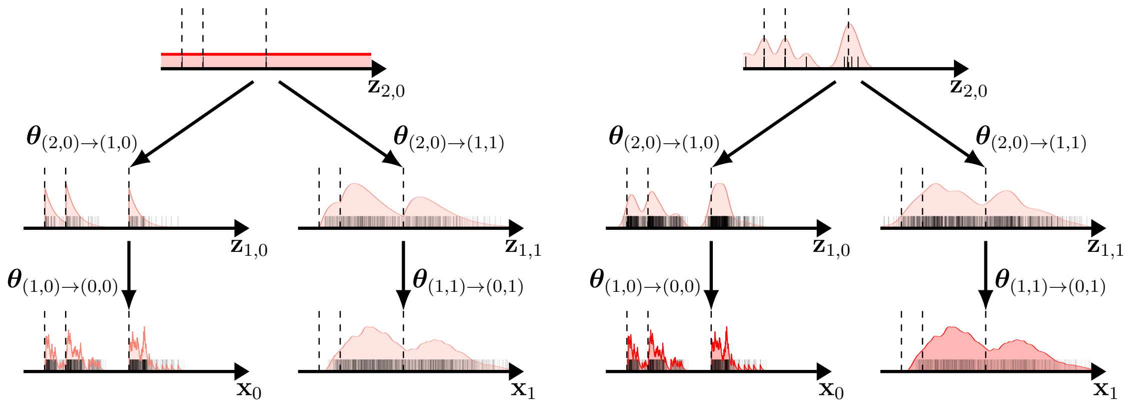


Figure 2.2: The distributions of the events for forward sampling (left) and posterior sampling (right). For forward sampling, three events are drawn from $\mathbf{Z}_{2,0}$. The dashed lines indicate the positions of the three events on the top layer. The plots for the other TPPs are the densities and rug plots of the samples drawn conditioned on $\mathbf{z}_{2,0}$. For posterior sampling, the samples for $\mathbf{x}_0$ and $\mathbf{x}_1$ are collected and fixed from the forward sampling. Then we draw posterior samples for the other TPPs conditional on $\mathbf{x}_0$ and $\mathbf{x}_1$ and plot their densities and rug plots as well. Note the modes of the posterior distribution for $\mathbf{z}_{2,0}$ recover the positions of the prior events for $\mathbf{z}_{2,0}$ in forward sampling.

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 gamma kernel that the gradients of the Weibull kernel function itself and the integral of the Weibull kernel function are both analytically available.

In this thesis, we mainly focus on DNSPs applied to TPPs, which we call temporal DNSPs, and the bounded region S is usually written as $[0, T]$, although many statements for TPPs are still true for general spatial DNSPs.

Chapter 3

Related Work

We list some related work in this chapter. Section 3.1 introduces the work using neural networks to model the TPPs. Section 3.2 describes the normalization-flow-based model for TPPs. Section 3.3 is about the Gaussian-process-related models for TPPs and SPPs. Section 3.4 is about the modeling technique based on piecewise constant function embedding. Section 3.5 reviews variational inference.

3.1 Neural Networks

Much recent work uses neural networks to directly model the CIF for a TPP (Du et al., 2016; Mei and Eisner, 2017; Zuo et al., 2020; Zhang et al., 2020).

Du et al. (2016) assume the CIF of a TPP is a nonlinear function as

$$\lambda(t) = \exp\left({\boldsymbol{v}^t}^{\mathrm{T}} \cdot \boldsymbol{h}_j + w^t(t - t_j) + b^t \right),$$

where $\boldsymbol{h}_j$ is the hidden state of the recurrent neural network (RNN) at j-th event, $\boldsymbol{v}^t$ is a column vector, w^t and b^t are scalars, and the probability for the $(j+1)$-th event to generate a mark k is

$$P(m_{j+1}=k|\boldsymbol{h}_j)=\frac{\exp\left(\boldsymbol{V}^y_{k,:}\boldsymbol{h}_j+b^y_k\right)}{\sum_{k=1}^K\exp\left(\boldsymbol{V}^y_{k,:}\boldsymbol{h}_j+b^y_k\right)},$$

where K is the number of marks, and $\boldsymbol{V}^y_{k,:}$ is the k-th row of matrix $\boldsymbol{V}^y$.

Mei and Eisner (2017) use a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Graves, 2012), a variant of an RNN, to model the CIF of what they call neural Hawkes processes. The CIF is

$$\lambda(t)=\sum_{k=1}^K\lambda_k(t)=\sum_{k=1}^K f_k(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{h}_j(t)),$$

where $f_k(x)=s_k\log\left(1+\exp\left(\frac{x}{s_k}\right)\right), s_k>0$. Different from Du et al. (2016), $\boldsymbol{h}_j(t)$ is continually obtained from memory cells and determined by the events up until time t_j. The probability of type k for the $(j+1)$-th event is

$$P(m_{j+1}=k|\boldsymbol{h}_j)=\frac{\lambda_k(t_{j+1})}{\lambda(t_{j+1})}.$$

Zhang et al. (2020) employ the self-attention mechanism to model the CIF. The functional form is

$$\lambda(t)=\sum_{k=1}^K\lambda_k(t)=\sum_{k=1}^K f(\mu_{k,j+1}+(\eta_{k,j+1}-\mu_{k,j+1})\exp(-\gamma_{k,j+1}(t-t_j))),$$

where

$$\mu_{k,j+1}=g(\boldsymbol{h}_{k,i+1}W_\mu),$$

$$\eta_{k,j+1}=g(\boldsymbol{h}_{k,j+1}W_\eta),$$

$$\gamma_{k,j+1}=f(\boldsymbol{h}_{k,j+1}W_\gamma),$$

f is the softplus function,

g is the Gaussian error linear unit for nonlinear activations,

$\boldsymbol{h}_{k,j+1}$ is the hidden vector learned by self-attention,

W_μ, W_η, W_γ are weight matrices.

The probability of type k for an event is the same as Mei and Eisner (2017).

Similar to Zhang et al. (2020), Zuo et al. (2020) build a transformer to learn a hidden state $\boldsymbol{h}_j$. They choose a functional form of the CIF similar to Du et al. (2016); Mei and Eisner (2017) as

$$\lambda(t) = \sum_{k=1}^{K} f_k \left(\alpha_k \frac{t - t_j}{t_j} + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{h}_j + b_k \right).$$

The probability of type k for an event is the same as Mei and Eisner (2017). While for the prediction task, Zuo et al. (2020) embed an extra layer for the time and type respectively.

Omi et al. (2019) propose to model the cumulative intensity function

$$\Lambda(\tau) = \int_0^\tau \lambda(t) dt$$

rather than the CIF. An RNN is utilized to learn the hidden state and then the hidden state and τ are fed into another cumulative intensity function network to output the cumulative intensity function. The direct modeling of the cumulative intensity function avoids the expensive numerical approximation of the integral.

Shchur et al. (2020a) develop a mixture model for TPPs. The $p.d.f$ used to describe the distribution of the inter-arrival time τ is

$$p(\tau | \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}) = \sum_{k=1}^{K} w_k \frac{1}{\tau s_k \sqrt{2\pi}} \exp \left(- \frac{(\log \tau - \mu_k)^2}{2 s_k^2} \right).$$

A history embedding given by a RNN, metadata and a sequence embedding are concatenated into a vector and fed into neural networks to output $\boldsymbol{w}, \boldsymbol{\mu}$ and $\boldsymbol{s}$.

All the above models learn the parameters by maximizing the log-likelihood function.

Mehrasa et al. (2019) try to use variational auto-encoder (VAE) (Kingma and Welling, 2014) to model the probability distributions of the inter-arrival time and the mark. The distribution of the inter-arrival time follows an exponential distribution:

$$p_\theta^\tau(\tau_j|z_j) = \lambda(z_j)\exp(-\lambda(z_j)\tau_j) \text{ for } \tau_j \geq 0.$$

The mark follows a multinomial distribution:

$$p_\theta^k(m_j = k|z_j) = p_k(z_j) \text{ and } \sum_{k=1}^{K} p_k(z_j) = 1.$$

$\lambda(z_j)$ and $p_\theta^k(m_j|z_j)$ are modeled using neural networks. The input sequence for the VAE is denoted as $x_{1:n} = \{x_1, \cdots, x_n\}$, where $x_n = (t_n, m_n)$, t_n is the time and m_n is the mark. Then the approximate posterior and prior of the latent state z_n are represented as multivariate Gaussian distributions:

$$q_\phi(z_n \mid x_{1:n}) = \mathcal{N}(\mu_{\phi_n}, \sigma^2_{\phi_n})$$

$$p_\phi(z_{n+1} \mid x_{1:n}) = \mathcal{N}(\mu_{\phi_{n+1}}, \sigma^2_{\psi_{n+1}}).$$

Two LSTMs are used to encode the history of events times and marks for prior and approximate posterior distributions. The hidden states of LSTMs are passed to extra networks to generate $\mu_{\phi_n}, \sigma^2_{\phi_n}, \mu_{\phi_n}, \sigma^2_{\phi_n}$. During the training process, the parameters are learned by optimizing the variational lower bound.

3.1.1 Summary

These methods train neural networks to learn the history embedding of TPPs. The use of GPUs (graphics processing units) makes it very efficient to train. The flexibility of neural networks makes the models fit different patterns of temporal data well.

However, all of these methods assume the latent space is deterministically identified by a neural network, which lacks the natural flexibility induced by the randomness of a stochastic process. Further, a huge number of parameters is generally required.

DNSPs assume the latent space is consisted of a random number of events, modeled by Poisson processes. The number of parameters used to control the kernels is far smaller than the number required by neural networks.

3.2 Normalization Flows

Shchur et al. (2020b) design a model, called TriTPP, for TPPs based on normalization flows. A triangular map $\boldsymbol{F}$ built with rational quadratic splines (RQSs) (Durkan et al., 2019) is used to model the cumulative intensity function.

The derivatives and inverse of $\boldsymbol{F}$ are in closed form. Thus, the sampling is fast by using the inversion method (Çinlar, 2013). The calculation of density is also fast as the CIF is just the derivative of $\boldsymbol{F}$ *w.r.t* the time. And $\boldsymbol{F}$ is highly flexible because of the flexibility of RQSs.

Similar to the models based on neural networks, TriTPP does not model the randomness mechanism. It assumes that there is a deterministic embedding based on RQSs for different sequences of data.

3.3 Gaussian Process (GP) Modulated Point Processes

Others have constructed GP-modulated point processes (Møller et al., 1998; Adams et al., 2009; Gunter et al., 2014; Lloyd et al., 2015; Lian et al., 2015; Donner and Opper, 2018; Aglietti et al., 2019). Suppose $f(t)$ is a realization of a Gaussian process, different choices of how to connect $\lambda(t)$ to $f(t)$ results in different models.

Møller et al. (1998) construct a log Gaussian Cox process (LGCP) whose intensity function is $\lambda(t) = \exp(f(t))$. An MCMC method is given to approximate the posterior intensity process.

The intensity function given by Adams et al. (2009) is $\lambda(t) = \lambda^* \sigma(f(t))$, where λ^* is an upper bound for $\lambda(t)$ and $\sigma(\cdot)$ is the sigmoid function. This kind of Cox process is called a sigmoidal Gaussian Cox process (SGCP). An MCMC method is also used to estimate the distribution of the posterior intensity function.

Gunter et al. (2014) build a dependent Cox point process. The latent processes are all Gaussian processes and a convolution is applied to each of the latent processes with a different kernel. Then the intensity function is the sum of all the latent processes after the convolution. Similar to Adams et al. (2009), an MCMC method called “adaptive thinning” is developed for the posterior inference, which uses a piecewise constant function to model the upper bound.

Lloyd et al. (2015) set the intensity function $\lambda(t) = f^2(t)$. The inference is performed by optimizing the evidence lower bound (ELBO). The Gaussian process f is dependent on a set of inducing points and the approximate distribution of the inducing points is assumed to be Gaussian.

Lian et al. (2015) combine a piecewise constant history embedding into a Gaussian process prior. The intensity function is $\lambda(t) = f^2(t)$, where $f(t)$ is a hierarchical Gaussian process. A low dimensional vector f_M is assumed to be from a Gaussian distribution and the $f(\cdot)$ as the input to $\lambda(\cdot)$ is a Gaussian process conditional on f_M. An approximate variational form of the Gaussian distribution for f_M is given and the inference is conducted through optimizing the ELBO.

Donner and Opper (2018) introduce a data augmentation scheme for the inference of SGCP. With the mean-field assumption, the inference can be done by optimizing an ELBO with an analytic form.

Aglietti et al. (2019) construct another data augmentation scheme for SGCP. By applying an augmentation of superposition of Poisson point processes, the variational inference avoids the calculation of numerical integrals and scales to large datasets well.

3.3.1 Summary

Such models assume the intensity functions are smooth. However, the intensity functions can experience sudden changes upon events arrivals, *e.g.*, a big earthquake can dramatically increase the probability to have a small earthquake in the near future. And the integral of the intensity function is not available in a closed form. Some approximations are required for the computation of the integral. Moreover, these models are not able to be stacked to construct a deep model built with point processes.

3.4 Piecewise-constant Intensity Model

Another line of work it to use a PCIM (Gunawardana et al., 2011) or other variants (Weiss and Page, 2013; Lian et al., 2015). Gunawardana et al. (2011) assume the conditional intensity function is piecewise constant. The intensity at each piece is assumed to be from a Gamma distribution and thus the posterior has a closed form. A decision-tree-based learning is developed for the learning of local structures. Weiss and Page (2013) extends the tree structures from Gunawardana et al. (2011) to the forests structures.

The history of events is embedded into a piecewise-constant function, which is the intensity function itself or a function that can output the intensity. The size of time windows for the calculation of the piecewise-constant function needs to be pre-determined. And these models lose the ability to have continuous history embedding.

3.5 Variational Inference

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, $D_{KL}(q \parallel p)$ (*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, $D_{KL}(p \parallel q)$. MSC uses MCMC to get an unbiased estimate of the gradient of the inclusive KL divergence. Naesseth et al. (2020) also show 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.

Chapter 4

Posterior Sampling

One of the fundamental questions we would like to ask is "What is the distribution of the points for the hidden layers given the observed data points?" Unfortunately, the posterior hidden point processes are not TPPs even when we only consider the temporal DNSPs with the kernels as in Example 2.4, because the events belonging to a point process have correlations to all the events from that point process and the events from other point processes connecting to that point process. The posterior sampling for a hidden point process is not trivial as it involves the sampling for an unbounded number of variables and there are no simple expressions for the posterior hidden point processes. In this chapter, we describe a posterior sampling method for temporal DNSPs, but the algorithm can be generalized to general spatial DNSPs easily.

The PCIF for our model is given in Proposition 4.1. See Appendix A for more details.

Proposition 4.1. *For temporal DNSPs, the Papangelou conditional intensity function for the posterior point process of $Z_{\ell,k}$ is*

$$\begin{aligned}\lambda_{\mathcal{P};\ell,k}(t) =& \lambda_{\ell,k}(t) \\ & \cdot \left(\prod_{i=1}^{K_{\ell-1}} \left(\exp\left(-\Phi_{(\ell,k)\to(\ell-1,i)}(T-t)\right) \prod_{t_{\ell-1,i,j}>t} \frac{\lambda'_{\ell-1,i}(t_{\ell-1,i,j},t)}{\lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right) \right),\end{aligned} \tag{4.1}$$

where

$$\lambda'_{\ell-1,i}(x,t) = \lambda_{\ell-1,i}(x) + \phi_{(\ell,k)\to(\ell-1,i)}(x-t)$$

and

$$\Phi_{(\ell,k)\to(\ell-1,i)}(x) = \int_0^x \phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(\tau)d\tau.$$

From Proposition 4.1, we can see that the PCIF at time t is not only controlled by the events on layers $\ell+1$ through $\lambda_{\ell,k}(\cdot)$, but controlled by the events on layers ℓ and $\ell-1$ through $\{\lambda_{\ell-1,i}(\cdot)\}_{i=1}^{K_{\ell-1}}$.

As the PCIF has complex correlations both within and between nodes, it is hard to directly do posterior sampling for the hidden point processes. Considering the simplest case, if we only have one hidden layer, our model just becomes a multivariate NSP. Typical methods for the posterior sampling of NSP include Metropolis-Hastings (M-H) algorithm (Metropolis et al., 1953; Norman and Filinov, 1969; Hastings, 1970), and spatial birth-and-death (SB&D) algorithm (Kelly and Ripley, 1976; Ripley, 1977; Baddeley and Møller, 1989). In the SPPs community, SB&D is more popular possibly due to its simplicity and efficiency (Geyer and Møller, 1994; Clifford and Nicholls, 1994; Møller and Waagepetersen, 2003). Unfortunately, the sufficient condition for the convergence of SB&D is not satisfied in our case, and it is an open question of whether SB&D converges for our model. We provide more details in Appendix B.

4.1 Markov Chain Monte Carlo

We devised a Markov chain equipped with auxiliary variables that converges quickly to the true posterior distribution. Compared with a naive MCMC sampler which just resamples all the hidden events from homogeneous Poisson processes every time as the proposal, our posterior sampling is much more efficient due to the help of auxiliary variables.

Remark 4.2. *Our MCMC can be applied to non-casual kernels (i.e., the kernels can be functions which have non-zero values in the whole space, like a Gaussian function) and SPPs with any dimensions, not only temporal DNSPs. No matter how we choose the kernel, the detailed balance and ergodicity conditions are still satisfied, and thus the MCMC sampler still converges to the posterior distribution.*

4.1.1 Virtual Events

Similar to prior work (Rao and Teh, 2011, 2013; Qin and Shelton, 2015; Shelton et al., 2018), we add virtual events as auxiliary variables for our MCMC sampler. They work by providing the candidates for the real events of the hidden point processes, and they do not contribute to any intensity functions. With the help of the virtual events, we only need to search for the real events where the virtual events appear, instead of the whole space. We explore all possible real event locations by resampling virtual events.

For each data point $\mathbf{x}$, there are L layers of virtual point processes (VPPs) $\tilde{\mathbf{Z}} = \{\tilde{\mathbf{Z}}_1, \ldots, \tilde{\mathbf{Z}}_L\}$ aligning with the hidden real point processes (RPPs) $\mathbf{Z}$, where $\tilde{\mathbf{Z}}_\ell = \{\tilde{Z}_{\ell,k}\}_{k=1}^{K_\ell}$

with $\tilde{Z}_{\ell,k}$ as a virtual point process. The CIF for $\tilde{Z}_{\ell,k}$ conditioned on $\mathbf{Z}_{\ell-1}$ is

$$\tilde{\lambda}_{\ell,k}(\tilde{t}) = \tilde{\mu}_{\ell,k} + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}} \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j} - \tilde{t}) \tag{4.2}$$

where $\tilde{\mu}_{\ell,k} \geq 0$ is the base rate, $\tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(\cdot)$ is the virtual kernel function, which we assume is a gamma kernel for TPPs as in Example 2.4. Note that $\tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j} - \tilde{t})$ evolves in the **opposite direction** to $\phi_{\boldsymbol{\theta}_{(\ell+1,i)\to(\ell,k)}}(t - t_{\ell+1,i,j})$, the reason for which we will explain later. However, the intensity of virtual events at layer ℓ depends on the **real events** at layer $\ell - 1$, not the virtual events there.

4.1.2 Complete Likelihood

The complete likelihood for the joint RPPs and VPPs for a data point is

$$p(\mathbf{x}, \mathbf{Z}{=}\mathbf{z}, \tilde{\mathbf{Z}}{=}\tilde{\mathbf{z}}) = p(\mathbf{z}_L) \prod_{\ell=0}^{L-1} p(\mathbf{z}_\ell \mid \mathbf{z}_{\ell+1}) \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_\ell) \tag{4.3}$$

where

$$p(\mathbf{z}_L) = \textstyle\prod_{k=1}^{K_L} p(z_{L,k}),$$

$$p(\mathbf{z}_\ell \mid \mathbf{z}_{\ell+1}) = \textstyle\prod_{k=1}^{K_\ell} p(z_{\ell,k} \mid \mathbf{z}_{\ell+1}),$$

$$\tilde{p}(\tilde{\mathbf{z}}_\ell \mid \mathbf{z}_{\ell-1}) = \textstyle\prod_{k=1}^{K_\ell} \tilde{p}(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1}),$$

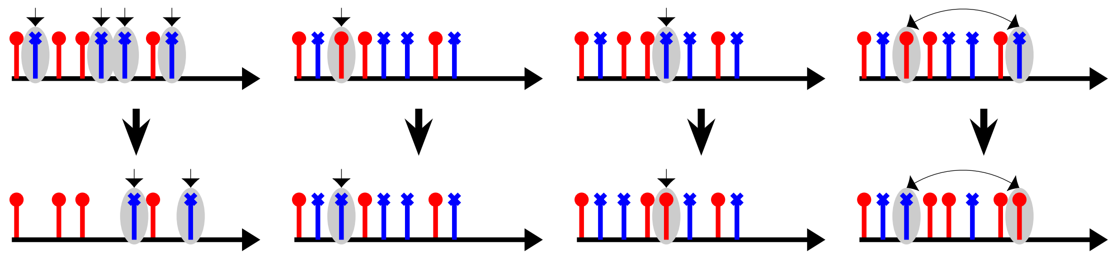
The likelihood of $z_{L,k}$ is

$$p(z_{L,k}) = \exp\left(-\mu_{L,k} T\right) \mu_{L,k}^{m_{L,k}},$$

where $m_{L,k}$ is the number of events drawn from $Z_{L,k}$.

The likelihood of $z_{\ell,k}$ for $0 \leq \ell \leq L-1$ is

$$p(z_{\ell,k} | \mathbf{z}_{\ell+1}) = \exp\left(\sum_{t_{\ell,k,j} \leq T} \log \lambda_{\ell,k}(t_{\ell,k,j}) - \int_0^T \lambda_{\ell,k}(t)\, dt \right).$$



(a) Re-sample virtual events (b) Flip: real to virtual (c) Flip: virtual to real (d) Swap

Figure 4.1: Examples for sampler moves.

(Here and for the rest of the paper, we let $z_{0,k}$ be $\mathbf{x}_k$.)

The likelihood of $\tilde{z}_{\ell,k}$ for $1 \leq \ell \leq L$ is

$$\tilde{p}(\tilde{z}_{\ell,k}|\mathbf{z}_{\ell-1}) = \exp\left(\sum_{\tilde{t}_{\ell,k,j} \leq T} \log \tilde{\lambda}_{\ell,k}(\tilde{t}_{\ell,k,j}) - \int_0^T \tilde{\lambda}_{\ell,k}(t)\, dt \right).$$

4.1.3 Sampler Moves

During the sampling process, we first select a hidden point process uniformly and then apply a move selected randomly from the following three types with a predetermined probability distribution. See Figure 4.1 for illustration, where ● represents a real event, and ✖ represents a virtual event.

Move 1: Re-sample virtual events. This move re-samples the virtual events for a VPP. The dimensionality of the samples is changed after each re-sampling. But the determinant of the Jacobian matrix, introduced as the correction for the changes of variables in reversible-jump MCMC (Green, 1995), is 1, since the new variables are independent of the current state of the Markov chain. The acceptance probability is always 1 for this move. See Appendix C.1 for more details.

Move 2: Flip. We uniformly pick an event from the union of the samples from the RPP and the VPP, and then propose to change the type for that event. If the type of the picked event is real, we propose to flip it to be a virtual event, and vice versa.

Move 3: Swap. One event is picked uniformly from the samples for each of the RPP and the VPP. Then we propose to swap the types of these two events, *i.e.*, the real event becomes a virtual event and vice versa.

Suppose the proposals for the changes are to adjust the events in $z_{\ell,k}$ and $\tilde{z}_{\ell,k}$ to become the events in $z'_{\ell,k}$ and $\tilde{z}'_{\ell,k}$. Then the likelihood ratio is

$$\mathcal{P} = \frac{p(z'_{\ell,k}|\mathbf{z}_{\ell+1})\tilde{p}(\tilde{z}'_{\ell,k}|\mathbf{z}_{\ell-1})}{p(z_{\ell,k}|\mathbf{z}_{\ell+1})\tilde{p}(\tilde{z}_{\ell,k}|\mathbf{z}_{\ell-1})} \cdot \frac{p(\mathbf{z}_{\ell-1}|\mathbf{z}'_{\ell})\tilde{p}(\tilde{\mathbf{z}}_{\ell+1}|\mathbf{z}'_{\ell})}{p(\mathbf{z}_{\ell-1}|\mathbf{z}_{\ell})\tilde{p}(\tilde{\mathbf{z}}_{\ell+1}|\mathbf{z}_{\ell})},$$

where

$$p(\mathbf{z}_{\ell-1} \mid \mathbf{z}_{\ell}) = \textstyle\prod_{k=1}^{K_{\ell-1}} p(z_{\ell-1,k} \mid \mathbf{z}_{\ell}),$$

$$\tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_{\ell}) = \textstyle\prod_{k=1}^{K_{\ell+1}} \tilde{p}(\tilde{z}_{\ell+1,k} \mid \mathbf{z}_{\ell}).$$

The ratio for the proposal probability is 1 for both *Move 2* and *Move 3*. So the acceptance probability for *Move 2* and *Move 3* is $\min(1, \mathcal{P} \cdot 1)$. See Appendix C.2 and C.3 for a detailed derivation.

It is necessary to have *Move 3* to help accelerate mixing even though *Move 2* seems to already include the ability to swap through two consecutive flips. However, often there is a real event that has large positive contribution to the likelihood in Equation 4.3. If we propose to flip this event to a virtual event, the likelihood ratio $\mathcal{P}$ would be very small, hence this real event would stay in the same place for a long time. Similarly, flipping a virtual event first would be unlikely.

With these three sampler moves, we can address why we use VPP intensity functions of the form of Equation 4.2:

1. The intensity functions for the VPPs are only controlled by the RPPs on the lower layers. Thus, we are able to sample the VPPs directly and efficiently, using the inversion method (Çinlar, 2013) in *Move 1*, to push the information from bottom to top.

2. The virtual events tend to appear more at the places where the probability is high to have a real event. Notice that the distribution of the real events as the children of the parent events on the upper layer follow the shape of the kernel functions. If the kernel function is decreasing, then the real events as children on the lower layers tend to be at places closer to the right of parent events on the upper layer. Then it makes sense to propose more virtual events on the upper layers closer to the left of real events on the lower layer. So it is natural to have virtual kernel functions that evolve with time in the reverse direction to the real kernel functions.

3. The base rates $\tilde{\mu}_{\ell,k}$ can help accelerate the mixing for the point processes not directly connected to the evidence. Consider a three-layer model with $\mathbf{x}$ as the evidence, $Z_{1,1}, Z_{2,1}$ as the RPPs, and $\tilde{Z}_{1,1}, \tilde{Z}_{2,1}$ as the VPPs. If there is no base rate $\tilde{\mu}_{2,1}$, then all the virtual events in $\tilde{Z}_{2,1}$ are drawn as the children of the real events in $Z_{1,1}$. And the virtual events from $\tilde{Z}_{2,1}$ are fixed when we propose to move the events belonging to $Z_{1,1}$. Then the probability to reject a *Move 2 or 3* for the events belonging to $Z_{1,1}$ would be very high, as the children from $\tilde{Z}_{2,1}$ would restrict the mobility of their

parents from $Z_{1,1}$. If instead we have a base rate $\tilde{\mu}_{2,1}$, then the base rate can help take care of the children and the parents have more freedom to move.

4.1.4 Update Virtual Kernels Parameters

Because we want the distribution of the proposed virtual events to be as close as possible to the posterior distribution of the real events, we would like to maximize the likelihood of the parameters for the VPPs assuming the posterior samples for the real events are drawn from the VPPs. That is, we use gradient ascent to adjust the parameters of the VPPs to make it more efficient (as the sampler is valid across different VPP parameters).

As the number of events usually varies significantly for different evidence samples, we assume the base rates $\tilde{\mu}_{n,L,i}$ and $\mu_{n,i}$ on the top layer are different for each data point $\mathbf{x}_n$, where n represents the data index.

Let $\mathbf{z}_n^{(1)}, \dots, \mathbf{z}_n^{(\mathcal{S})}$ be the posterior samples from the distribution $p(\mathbf{Z}_n \mid \mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\theta})$, where $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_{(\ell+1,*)\to(\ell,*)}\right]_{\ell=0}^{L-1}$ are the parameters of the kernel functions and $\boldsymbol{\mu}_n = [\mu_{n,i}]_{i=1}^{K_L}$ are the parameters of the HPPs on the top layer. Given the log-likelihood of the parameters of the VPPs *w.r.t* the real events

$$\tilde{\text{llh}}\left(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\mu}}_n; \mathbf{x}_n, \mathbf{z}_n\right) = \sum_{\ell=1}^{L} \tilde{\text{llh}}_{(\ell-1,*)\to(\ell,*);\mathbf{x}_n,\mathbf{z}_n}$$

where

$$\tilde{\text{llh}}_{(\ell-1,*)\to(\ell,*);\mathbf{x}_n,\mathbf{z}_n} = \sum_{k=1}^{K_\ell} \left(\sum_{t_{n,\ell,k,j} \leq T} \log \tilde{\lambda}_{n,\ell,k}(t_{n,\ell,k,j}) - \int_0^T \tilde{\lambda}_{n,\ell,k}(t)dt \right),$$

the update rules are

$$\tilde{\boldsymbol{\mu}}_n \leftarrow \tilde{\boldsymbol{\mu}}_n + \frac{\tilde{r}}{\mathcal{S}N} \sum_{s=1}^{\mathcal{S}} \nabla_{\tilde{\boldsymbol{\mu}}_n} \tilde{\text{llh}} \left(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\mu}}_n; \mathbf{x}_n, \mathbf{z}_n^{(s)} \right), \tag{4.4}$$

$$\tilde{\boldsymbol{\theta}} \leftarrow \tilde{\boldsymbol{\theta}} + \frac{\tilde{r}}{\mathcal{S}N} \sum_{n=1}^{N} \sum_{s=1}^{\mathcal{S}} \nabla_{\tilde{\boldsymbol{\theta}}} \tilde{\text{llh}} \left(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\mu}}_n; \mathbf{x}_n, \mathbf{z}_n^{(s)} \right), \tag{4.5}$$

where

$$\tilde{\boldsymbol{\mu}}_n = \left[[\tilde{\mu}_{n,\ell,k}]_{\ell=1}^{L} \right]_{k=1}^{K_\ell},$$

$$\tilde{\boldsymbol{\theta}} = \left[\left[\left[\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,j)} \right]_{i=1}^{K_{\ell-1}} \right]_{j=1}^{K_\ell} \right]_{\ell=1}^{L},$$

N is the number of data points, and $\tilde{r} > 0$ is the step size for optimizing. See Appendix D.1 for the gradient.

Chapter 5

Inference

In this chapter, we introduce moment-based inference in Section 5.1, Bayesian inference in Section 5.2, and Monte Carlo expectation-maximization in Section 5.3. Moment-based inference chooses to minimize the contrast between a kernel estimation and the theoretical expression of the pair correlation function. Moment-based inference does not involve the simulation of the posterior hidden point processes and thus is computationally efficient. Bayesian inference and Monte Carlo expectation-maximization all require posterior samples of the posterior hidden point processes. Bayesian inference produces the samples of the parameters from the posterior distributions of the parameters, while Monte Carlo expectation-maximization estimates the set of parameters by maximizing the marginal likelihood of the observed data. We also investigate variational inference with exclusive Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951; Kullback, 1997) in Section 5.4 and inclusive KL divergence in Section 5.5.

5.1 Moment-based Inference

Andersen et al. (2018) give a moment-based method for the inference of univariate NSPs of order 2, but it could be easily generalized to univariate NSPs with any order. Although the moment-based inference is computationally easy, some hyper-parameters for the estimation of the moments have to be tuned manually and it is not able to estimate the posterior intensity surface of the unobservable point processes.

5.2 Bayesian Inference

We use MCMC to approximate the posterior distributions of the parameters. We use $\mathbf{\Theta} = \{\boldsymbol{\theta}, \{\boldsymbol{\mu}_n\}\}$ to denote all of the parameters. We need to sample $\mathbf{\Theta}$ from the posterior distribution

$$p(\mathbf{\Theta} \mid \mathbf{x}, \mathbf{z}) = \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{\Theta})}{p(\mathbf{x}, \mathbf{z})}.$$

Each time we uniformly select a parameter Θ from $\mathbf{\Theta}$. The acceptance ratio is

$$\mathcal{A} = \min(\alpha, 1),$$

where

$$\alpha = \frac{p(\Theta' \mid \mathbf{x}, \mathbf{z}) q(\Theta' \to \Theta)}{p(\Theta \mid \mathbf{x}, \mathbf{z}) q(\Theta \to \Theta')} = \frac{p(\Theta') p(\mathbf{x}, \mathbf{z} \mid \Theta') q(\Theta' \to \Theta)}{p(\Theta) p(\mathbf{x}, \mathbf{z} \mid \Theta) q(\Theta \to \Theta')},$$

Θ' is the proposes value, and Θ is the current value.

We assume the prior distributions for each parameter is Gamma distribution, and use h to denote the shape, c to denote the scale:

$$p(\Theta) = \text{Gamma}(\theta; h, c),$$

$$q(\Theta' \to \Theta) = \text{Gamma}(\Theta; h, \Theta'/h),$$

$$q(\Theta \to \Theta') = \text{Gamma}(\Theta'; h, \Theta/h),$$

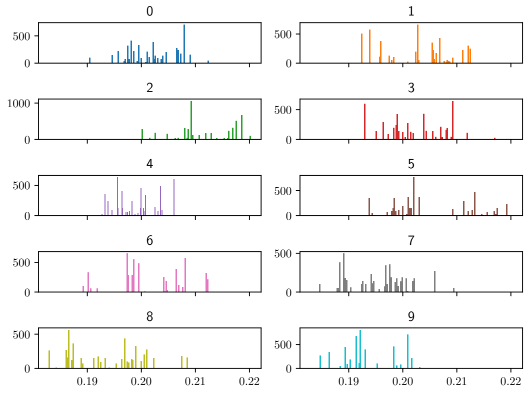
and

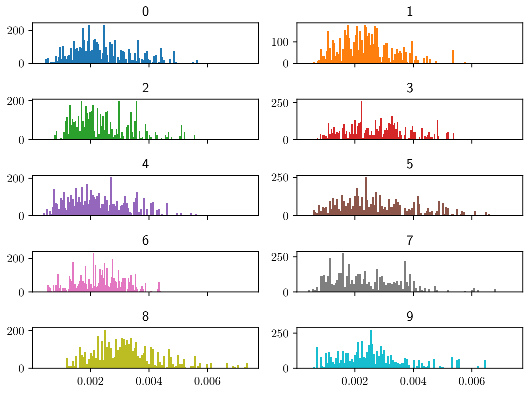
$$\begin{aligned}\alpha =& \frac{\cancel{1/(\Gamma(h)c^h)(\Theta')^{h-1}} \exp(-\Theta'/c) \cdot 1/\left(\cancel{\Gamma(h)}(\Theta'/h)^h\right) \cancel{\Theta^{h-1}} \exp(-\Theta/(\Theta'/h)) \cdot p(\mathbf{x}, \mathbf{z} \mid \Theta')}{\cancel{1/(\Gamma(h)c^h)\Theta^{h-1}} \exp(-\Theta/c) \cdot 1/\left(\cancel{\Gamma(h)}(\Theta/h)^h\right) \cancel{(\Theta')^{h-1}} \exp(-\Theta'/(\Theta/h)) \cdot p(\mathbf{x}, \mathbf{z} \mid \Theta)} \\ =& \frac{\Theta^h \exp(-\Theta'/c - \Theta/(\Theta'/h)) \cdot p(\mathbf{x}, \mathbf{z} \mid \Theta')}{(\Theta')^h \exp(-\Theta/c - \Theta'/(\Theta/h)) \cdot p(\mathbf{x}, \mathbf{z} \mid \Theta)}.\end{aligned}$$

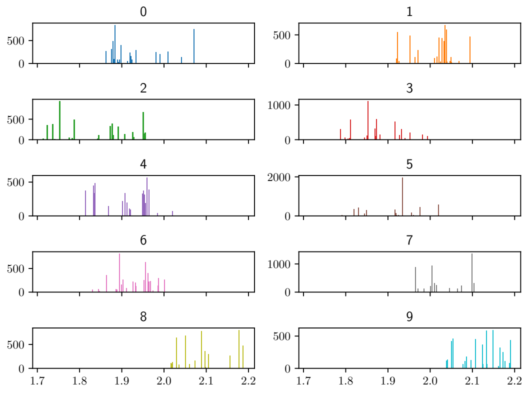
We generate synthetic data and do posterior sampling for both the parameters and the hidden events for a DNSP with 2 hidden layers and 1 type. The kernel function is an exponential kernel function (*i.e.*, $\phi_{\boldsymbol{\theta}}(x) = \mu + \alpha\beta \exp(-\beta x)$). We collect samples for parameters as shown in Figure 5.1. After 10000 burn-in steps, we collect 10 sets of samples with sample size as 5000 for each parameter. μ_0 is the constant rate for the top layer, μ_1, α_1, β_1 are for layer 1, and μ_2, α_2, β_2 are for layer 2.

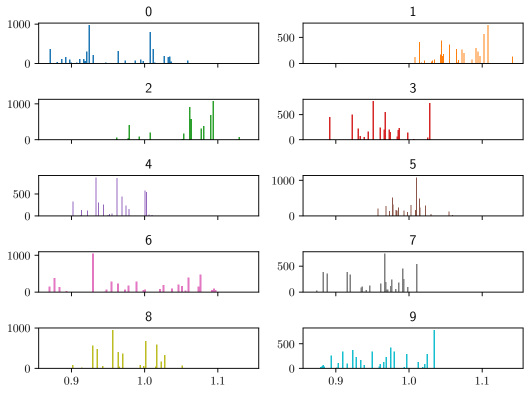
5.3 Monte Carlo Expectation-Maximization

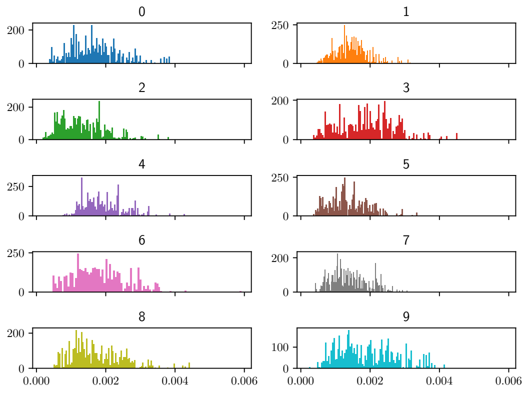
Bayesian inference requies the posterior sampling for the parameters, but in our case it becomes too slow to be feasible if we have many parameters. For more efficient inference, we adopt Monte Carlo expectation-maximization (MCEM) (Wei and Tanner, 1990) as an indirect way to maximize the marginal likelihood. Different from the ordinary


(a) μ_0, shape=4, scale=0.05


(b) μ_1, shape=4, scale=0.0005


(c) α_1, shape=4, scale=0.5


(d) β_1, shape=4, scale=0.25


(e) μ_2, shape=4, scale=0.0005

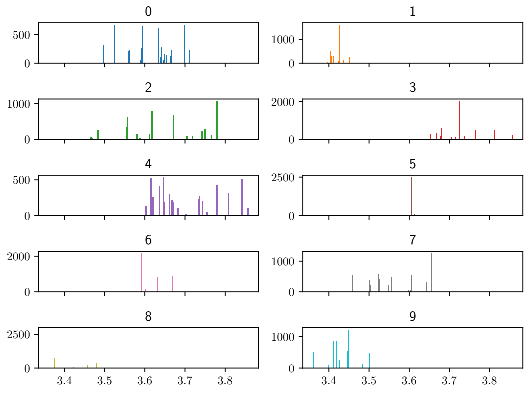
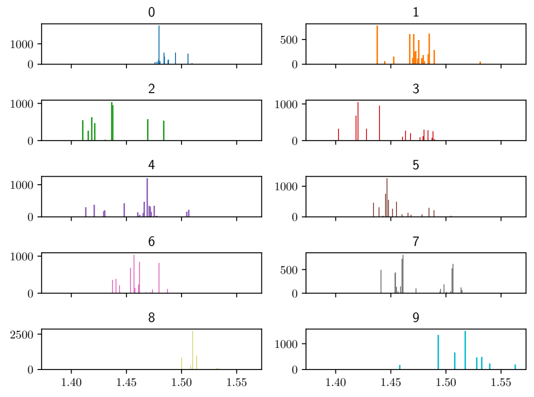

(f) α_2, shape=4, scale=0.875

Figure 5.1: Full Bayesian Inference for Parameters



(g) β_2, shape=4, scale=0.375

Figure 5.1: Full Bayesian Inference for Parameters (cont.)

EM, the expected value of the log-likelihood is approximated by a Monte Carlo method in the expectation steps. Posterior samples of the hidden point processes are required for the estimation of the expected values.

MCEM iteratively applies the following two-step process until the parameters for the RPPs converge.

Step 1. Generate posterior samples $\mathbf{z}_n^{(1)}, \ldots, \mathbf{z}_n^{(\mathcal{S})}$ from the distribution $p(\mathbf{Z}_n \mid \mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\theta})$.

Step 2. Update the parameters. Given the log-likelihood function of the parameters of the RPPs *w.r.t* the real events

$$\text{llh}(\boldsymbol{\theta}, \boldsymbol{\mu}_n; \mathbf{x}_n, \mathbf{z}_n) = \log \left(\prod_{i=1}^{K_L} p(z_{n,L,i}) \cdot \prod_{\ell=0}^{L-1} \prod_{i=1}^{K_\ell} p(z_{n,\ell,i} \mid \mathbf{z}_{n,\ell+1}) \right), \tag{5.1}$$

the update rules for the parameters are

$$\boldsymbol{\mu}_n \leftarrow \arg\max_{\boldsymbol{\mu}_n} \left\{ \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} \text{llh}(\boldsymbol{\theta}, \boldsymbol{\mu}_n; \mathbf{x}_n, \mathbf{z}_n^{(s)}) \right\}, \tag{5.2}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \frac{r}{\mathcal{S}N} \sum_{n=1}^{N} \sum_{s=1}^{\mathcal{S}} \nabla_{\boldsymbol{\theta}} \mathrm{llh}(\boldsymbol{\theta}, \boldsymbol{\mu}_n; \mathbf{x}_n, \mathbf{z}_n^{(s)}), \tag{5.3}$$

where N is the number of data points and $r > 0$ is the step size for optimizing. See Appendix D.2 for the gradient and maximization formula.

The inference here is not the standard MCEM. We combine full maximization (Equation 5.2) and ascent-based MCEM (Equation 5.3) described in Caffo et al. (2005). When the sample size $\mathcal{S}$ goes to infinity and we update the parameters as in Equations 5.2 and 5.3, the expected value of the log-likelihood in Equation 5.1 increases at each iteration with probability converging to 1 (Caffo et al., 2005). The general theory for the convergence of MCEM has not been well-established. Different senses and different approaches for the convergence analysises are given by Neath et al. (2013). We tried various settings for MCEM and choose the one that is stable and computationally efficient. Adam (Kingma and Ba, 2015) was utilized for the optimization of the parameters of the kernel and virtual kernel functions. Notice that we are not changing the parameters for the real or virtual kernels during MCMC sampling. The posterior samples for the real events are collected from the MCMC sampler after it reaches convergence. Pseudo-code is given in Algorithm 1.

5.3.1 Experiments

The code is available online at `https://github.com/hongchengkuan/Deep-Neyman-Scott-Processes`.

Algorithm 1 MCEM for DNSPs

```
Input: data $\{\mathbf{x}_n\}$, model $\mathcal{M}$
Initialization: base rates $\{\boldsymbol{\mu}_n\}$, kernel parameters $\boldsymbol{\theta}$, virtual base rates $\{\tilde{\boldsymbol{\mu}}_n\}$, virtual kernel parameters $\tilde{\boldsymbol{\theta}}$, initial states for the Markov chains.
1: for $n = 1$ to $N$ do
2:     repeat
3:         $\mathbf{z}_n^{(1)}, \dots, \mathbf{z}_n^{(\mathcal{S})} \sim p(\mathbf{Z}_n \mid \mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\theta})$ by MCMC
4:         Maximize base rates based on Equation 5.2    ▷ Maximize the marginal likelihood
5:         Use Adam to optimize Equation 5.3            ▷ Maximize the marginal likelihood
6:         Use Adam to optimize Equations 4.4 and 4.5
7:         Record $\mathbf{z}_n^{(\mathcal{S})}$ as the initial state for the next Markov chain in MCMC
8:     until the expected log-likelihood in Equation 5.1 converges
9: end for
```

Architectures

We constructed hierarchical models as in Fig. 5.2. Black horizontal arrows are point processes. Gray arrows are model connections. For the 1-hidden model, we only have one layer of hidden TPPs and there is only one TPP in total on the top layer. The hidden TPP is connected to all the types of events in the evidence.

For the 2-hidden model, we have the one hidden TPP on layer 1 for each type in the evidence with a single connection between matched hidden- and observed-TPPs, *i.e.*, $\phi_{\boldsymbol{\theta}_{(1,i)\to(0,j)}}(\cdot) = 0$ if $i \neq j$. There is also only one hidden TPP on the top layer, which is connected to all hidden TPPs on layer 1.

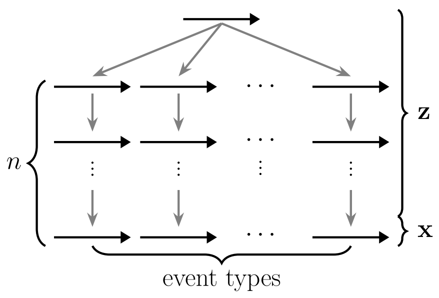


Figure 5.2: n-hidden

The n-hidden model can be constructed by adding more hidden layers for each type similarly. For each dataset, the **best** result is shown in bold and underlined, the **runner-up** in bold.

Training and Testing

The probabilities of the moves for *Move 1*, *Move 2*, and *Move 3* are 0.2, 0.6, and 0.2 respectively. We train the models using Algorithm 1 to get the parameters of the kernel and virtual kernel functions.

During testing, the parameters of the kernel functions and virtual kernel functions are fixed. We update the base rates according to Eqs. 5.2 and 4.4. The posterior samples for RPPs are collected to calculate the expectation of the log-likelihood per event. For each sequence in the evidence, there is an event at the end. To better capture the last event, we assume there is a synthetic real event for each hidden TPP at the end. This synthetic real event is only involved with the calculation of the intensity functions for the VPPs.

Likelihood

For a neural network, the CIF has a parametric form $\lambda_{f_\Theta(\mathbf{x})}(t)$ determined by a function f_Θ and the data $\mathbf{x}$. The parameters Θ for f_Θ are learned during training. For testing, the CIF $\lambda_{f_\Theta(\mathbf{x})}(t)$ is determined by the testing data $\mathbf{x}$ and f_Θ. The log-likelihood is $\log P(\mathbf{x} \mid \lambda_{f_\Theta(\mathbf{x})}(t))$.

For a DNSP, Θ, the parameters for the kernels, are learned during training, and fully specify $P(\mathbf{Z} \mid \mathbf{x})$. We use a random function f_Θ to determine the CIF. The distribution of $f_\Theta(\mathbf{x})$ is fully determined by $P(\mathbf{Z} \mid \mathbf{x})$, and thus is fully determined by Θ and $\mathbf{x}$. We calculate the expected value of the log-likelihood $\mathbb{E}_{f_\Theta(\mathbf{x})}[\log P(\mathbf{x} \mid \lambda_{f_\Theta(\mathbf{x})}(t))] = \mathbb{E}_{\mathbf{Z}\sim P(\mathbf{Z}|\mathbf{x})}[\log P(\mathbf{x} \mid \mathbf{Z})]$ to compare with the baselines.

We do not compare the marginal log-likelihood $\log \mathbb{E}_{\mathbf{Z}\sim P(\mathbf{Z})}[P(\mathbf{x} \mid \mathbf{Z})]$ of our model with the log-likelihood of the neural-network-based models because there are no efficient methods for estimating this expectation, as "forward sampling" with $P(\mathbf{Z})$ is prohibitively inefficient and samples from $P(\mathbf{Z} \mid \mathbf{x})$, which our method is designed to generate, cannot be used (as the importance sampling ratio weight cannot be calculated).

Similarly, it is difficult to calculate the marginal likelihood for MTPP and the log-likelihood for MTPP is the evidence lower bound per event.

To address the difference in likelihood calculations across methods, we also provide two additional metrics: predictive accuracy and root mean squared error, which are handled in exactly the same fashion for all methods (see below).

Prediction

For prediction, each method predicts the time and type of the next event, conditioned on all events prior to it. This process is repeated for every event in the testing data (slowly increasing the conditioning set for each new prediction).

Instead of calculating an infinite integral as in NHP and SAHP, or using an additional layer of a neural network as in THP, we simply do forward sampling to predict the time and type for the next future event, as we have an explicit formula for the intensity.

We predict the time for the next future event from the beginning to the end. After we get to the convergence of our Markov chain, we draw the samples for the next future event $e_i = (t_i, k_i)$ conditional on the current state of the Markov chain. Suppose the samples for the next future event e_i conditional on the history $\mathcal{H}_{i-1} = \{(t_1, k_1), (t_2, k_2), (t_3, k_3), \cdots, (t_{i-1}, k_{i-1})\}$ are $(t_i^1, k_i^1), (t_i^2, k_i^2), \ldots, (t_i^{\mathcal{S}}, k_i^{\mathcal{S}})$, where $\{t_i^j\}_{j=1}^{\mathcal{S}}$ are the times of the samples for the future event e_i and $\{k_i^j\}_{j=1}^{\mathcal{S}}$ are the types of the future event e_i. The prediction for the future event time is $\hat{t}_i = \frac{1}{\mathcal{S}} \sum_{j=1}^{\mathcal{S}} t_i^j$ and the type prediction is $\hat{k}_i = \arg\max_{k \in \{1,\ldots,K_0\}} \sum_{j=1}^{\mathcal{S}} \mathbb{1}_k(k_i^j)$, where $\mathbb{1}_k(k_i^j)$ is the indicator function which is equal to 1 *iff* $k = k_i^j$. Then we calculate the root mean squared error (RMSE) for the time prediction and accuracy for the type prediction.

After the prediction for e_i, we use the current state of the Markov chain for $\mathcal{H}_{i-1}$ as the initial state for the Markov chain for $\mathcal{H}_i$ and run the Markov chain until convergence. Then we predict the time and type for the event e_{i+1} conditional on $\mathcal{H}_i$ the same as how we predict the event e_i. The pseudocode can be found in Algorithm 2.

Algorithm 2 Predict e_{n+1} conditional on $\{e_1, e_2, \cdots, e_n\}$

```
Input: observed data $\mathbf{x} = \{e_1, e_2, \cdots, e_n\}$, where $e_i = (t_i, k_i)$, and model $\mathcal{M}$
Initialization: $\mathbf{\Theta}$, $\tilde{\mathbf{\Theta}}$, and sample size $\mathcal{S}$.
Output: time prediction $\hat{t}_{n+1}$, type prediction $\hat{k}_{n+1}$
1: for $s = 1$ to $\mathcal{S}$ do
2:     sample $\mathbf{z}^s \sim p(\mathbf{z} \mid \mathbf{x}; \mathbf{\Theta})$        ▷ posterior sampling
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 $\mathbf{z}^s \sim p(\mathbf{z} \mid \mathbf{x}; \mathbf{\Theta})$        ▷ posterior sampling
7:     sample the future time $\hat{t}^s_{n+1}$ based on $\mathbf{z}^s$        ▷ time extension sampling
8:     sample the future type $\hat{k}^s_{n+1}$ based on $\mathbf{z}^s$        ▷ time extension sampling
9: end for
10: $\hat{t}_{n+1} = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} \hat{t}^s_{n+1}$
11: $\hat{k}_{n+1} = \underset{k \in \{1,\ldots,K_0\}}{\arg\max} \sum_{s=1}^{\mathcal{S}} \mathbb{1}_k(\hat{k}^s_{n+1})$
```

Synthetic Data Experiments

We use synthetic data to illustrate the modeling power of multiple layers. The synthetic data are generated from DNSPs with differing numbers of hidden layers and 2 event types. They are divided into training and test sets. Then we apply our models with different number of hidden layers to train and test the synthetic datasets. The log-likelihood per event, time prediction root mean squared error (RMSE) and type prediction accuracy for the test set are shown in Table 5.1. The leftmost column represents the different depths we use to generate synthetic data and the right 3 columns are the results when we use different model depths for training and testing. For log-likelihood (which our model is trained for), increasing depth helps, particularly up to the depth of the data. Accuracy and RMSE have similar behavior.

Table 5.1: Synthetic Experimental Results

Synthetic Datasets	Model	Log-likelihood	RMSE	Accuracy
2-hidden	1-hidden	-0.006	1.048	0.722
	2-hidden	0.286	$\underline{\mathbf{0.942}}$	**0.760**
	3-hidden	**0.301**	**1.010**	$\underline{\mathbf{0.762}}$
	4-hidden	$\underline{\mathbf{0.304}}$	1.189	0.757
3-hidden	1-hidden	0.528	0.731	0.782
	2-hidden	**0.822**	$\underline{\mathbf{0.611}}$	**0.812**
	3-hidden	$\underline{\mathbf{0.835}}$	**0.613**	$\underline{\mathbf{0.814}}$
	4-hidden	0.831	0.670	0.809
4-hidden	1-hidden	1.177	0.411	0.820
	2-hidden	1.458	0.409	0.845
	3-hidden	**1.464**	$\underline{\mathbf{0.391}}$	$\underline{\mathbf{0.846}}$
	4-hidden	$\underline{\mathbf{1.466}}$	$\underline{\mathbf{0.391}}$	$\underline{\mathbf{0.846}}$

Real-world Data Experiments

We compare our DNSP to four currently popular continuous-time event modeling methods: MTPP (Lian et al., 2015), the neural Hawkes process (NHP) (Mei and Eisner, 2017), the self-attentive Hawkes process (SAHP) (Zhang et al., 2020), and the transformer Hawkes process (THP) (Zuo et al., 2020). MTPP combines Gaussian processes and piecewise-constant intensity models (Gunawardana et al., 2011), while the other baselines are based on neural networks. The datasets we use are retweets, earthquakes, and homicides. Each dataset is split into training, validation, and test sets randomly five times. We report the mean values with the standard deviations shown in parentheses. Validation is used to tune the hyperparameters and early stopping for all the datasets for NHP, SAHP and THP. For DNSPs, validation is only used for early stopping for retweets dataset, as we only use mini-batch gradient ascent for retweets. Batch gradient ascent and termination based on training are used for earthquakes and homicides. For MTPP, each task corresponds to a type in our experiments. The model parameters of MTPP are fixed across different sequences and each sequence has their own variational parameters. We use forward sampling to do the prediction for MTPP in a similar way.

Datasets

Retweets (Zhao et al., 2015) The retweets dataset collected sequences of tweets streams. Each sequence contains the times and types for some follow-up retweets. We use the same dataset as used in NHP. The retweets are grouped into three types (small, medium and large) according to the number of followers of the users who owned the retweets.

Earthquakes (NCEDC, 2014; BDSN, 2014; HRSN, 2014; BARD, 2014) We collected the times and magnitudes for earthquakes between 01/01/2014 00:00:00 and 01/01/2020 00:00:00 in the region spanning between 34.5° and 43.2° latitude and between $-126.00°$ and $-117.76°$ longitude. If the magnitude of a earthquake is smaller than 1, we classify it as a small earthquake, otherwise a large earthquake.

Homicides (COC) This dataset contains the times for homicides that occurred at five contiguous districts (007-011) with the most homicides in Chicago from 01/01/2001 00:00:00 to 01/01/2020 00:00:00. The type for an event is the district where the homicide occurred. The terms of use can be found at `https://www.chicago.gov/city/en/narr/foia/data_disclaimer.html`.

The number of types, the number of total events and the number of sequences for each dataset are summarized in Table 5.2.

Table 5.2: Datasets Statistics

Datasets	# types	# events	# sequences		
			train	validation	test
Retweets	3	2610102	20000	2000	2000
Earthquakes	2	156743	209	53	53
Homicides	5	3956	6	2	2

Results The results in Table 5.3 indicate that our model achieves competitive results compared to the baselines. We have the best RMSE for earthquakes and homicides, the best accuracy for earthquakes, and the best likelihood for retweets and homicides. The likelihood for the earthquakes dataset is only a little worse than SAHP and THP. Note that *our model only needs to fit tens of parameters during training, compared to the hundreds of thousands parameters needed for the neural networks.*

It is also notable that the 2-hidden is better than 1-hidden in terms of likelihood and accuracy while, for RMSE, 2-hidden is worse than 1-hidden consistently. It indicates that separate hidden TPPs for each type does increase the power to fit the evidence. The RMSE of the 1-hidden model is not maintained when increasing model capacity to the 2-hidden model, because we are fitting to likelihood which is related but different than RMSE. Note that on the homicides data, our RMSE error for both 2-hidden and 1-hidden are significantly better than those of prior work.

We also ran the real-world experiments for both a fully connected model and a model with more than 2 hidden layers. The likelihood, RMSE, and accuracy were not improved, because the single top layer is sufficient to correlate events across the different types.

Time Complexity The time complexities for flip, swap, and resampling for a single event in MCMC are $\mathcal{O}(1)$, constant irrespective of any values (amount of data, parameter values, length of time, etc). A full analysis of the MCMC time complexity would require bounding

Table 5.3: Real-World Experimental Results

Datasets	Model	Log-likelihood	RMSE($\times 10^4$)	Accuracy
	MTPP	$-13.44(0.18)$	1.64(0.03)	0.36(0.00)
	NHP	$-6.12(0.03)$	1.64(0.03)	0.48(0.01)
Retweets	SAHP	$-4.38(0.17)$	**1.60(0.04)**	0.51(0.03)
	THP	$-4.63(0.03)$	**1.54(0.03)**	**0.61(0.00)**
	1-hidden	**$-4.15(0.07)$**	**1.60(0.03)**	0.48(0.00)
	2-hidden	**$-3.54(0.15)$**	1.69(0.07)	**0.57(0.00)**
	MTPP	$-10.45(0.09)$	0.20(0.01)	0.52(0.00)
	NHP	$-8.70(0.08)$	0.20(0.01)	0.39(0.01)
Earthquakes	SAHP	**$-8.29(0.25)$**	**0.16(0.01)**	0.56(0.03)
	THP	**$-8.13(0.05)$**	0.20(0.01)	**0.61(0.01)**
	1-hidden	$-8.45(0.05)$	**0.15(0.01)**	**0.61(0.01)**
	2-hidden	$-8.43(0.06)$	**0.16(0.01)**	**0.60(0.01)**
	MTPP	$-18.15(1.40)$	23.14(3.41)	0.20(0.02)
	NHP	$-21.93(0.93)$	23.13(3.42)	0.18(0.03)
Homicides	SAHP	$-14.02(0.33)$	23.13(3.42)	0.19(0.02)
	THP	$-14.87(0.63)$	23.13(3.42)	**0.26(0.03)**
	1-hidden	**$-11.68(0.20)$**	**17.40(2.68)**	**0.25(0.01)**
	2-hidden	**$-10.78(0.11)$**	**17.73(2.84)**	**0.25(0.02)**

the number of steps necessary (by mixing time arguments, usually). As with almost all other non-trivial MCMC inference methods, we do not have such a bound.

Intuitively, a model with more hidden layers has much larger search space for the events from a posterior distribution and it should take more time to get fully mixed. And we have observed experimental results to support our intuition. As shown in Table 5.4, the 2-hidden models take more time to get fully mixed than 1-hidden models.

The experiments are trained on a cluster with multiple CPU cores. The number of CPU cores ranges from 8 to 64.

Table 5.4: Training Time in Hours

Datasets	Model	Time
Retweets	1-hidden	12.0
	2-hidden	108.5
Earthquakes	1-hidden	0.8
	2-hidden	22.3
Homicides	1-hidden	1.1
	2-hidden	14.0

5.4 Variational Inference with Exclusive KL Divergence

Although the MCMC algorithm can approximate the true posterior distribution with an arbitrarily small error, the burn-in steps usually consume a lot of time. Moreover, for every sequence of data, we need to maintain different real and virtual events sets with different number of events and there is no single instruction we can use to manipulate all the events sets at the same time, thus the GPU implementation of MCMC is not immediately available. Variational inference (VI) (Jordan et al., 1999) casts the problem of approximating the posterior distribution problem into an optimization problem. For every sequence of data, we use a same number of parameters to describe the hidden point processes and the sequences of events from a same point process can be padded with dummy values into sequences with a same length.

For DNSPs, the approximate distribution can break up the correlations between hidden points and assume the hidden points are independently distributed. In doing so, the hidden points are completely characterized by an intensity function, which specifies the probability of the number of hidden points at each infinitesimal region as a Bernoulli random variable. The intensity function is then obtained by minimizing the exclusive Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951; Kullback, 1997) between our assumed distribution and the posterior distribution of hidden points.

Definition 5.1 (KL divergence)**.** *The general KL divergence from a point process with probability measure P to a point process with probability measure Q is*

$$D_{KL}(Q \parallel P) = \int \log\left(\frac{dQ}{dP}\right) dQ.$$

The approximate point processes for the posterior point processes are independent Poisson processes $\mathbf{Z}^E = \{\mathbf{Z}_1^E, \cdots, \mathbf{Z}_L^E\}$, with $\mathbf{Z}_\ell^E = \{Z_{\ell,k}^E\}_{k=1}^{K_\ell}$. The intensity function for $Z_{\ell,k}^E$ is assumed to be $\lambda_{\ell,k}^E(\cdot)$. Under this assumption, all of the correlations between hidden events are removed.

Proposition 5.2. *Minimizing the exclusive KL divergence $D_{KL}(Q(\mathbf{Z}^E) \parallel P(\mathbf{Z}))$ is equivalent to maximizing the evidence lower bound (ELBO)*

$$
\begin{aligned}
\mathcal{L} = &-\sum_{\ell,i} \int_{S_{\ell,i}} \lambda^E_{\ell,i}(t)(\log \lambda^E_{\ell,i}(t) - 1)dt \\
&+ \Bigg(\sum_{t^x_{0,k,j}} \int \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h}) dQ(\mathbf{Z}^E) \\
&\quad - \sum_{\ell\geq 1,i,k} \int_{S_{\ell,i}} \Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}}(t)\lambda^E_{\ell,i}(t)dt \\
&\quad + \sum_{\ell\geq 1} \int \sum_{k,j} \log \sum_{i,h} \phi_{(\ell+1,i)\to(\ell,k)}(t^z_{\ell,k,j} - t^z_{\ell+1,i,h}) dQ(\mathbf{Z}^E) \Bigg),
\end{aligned}
\tag{5.4}
$$

where $\{S_{\ell,i}\}$ are the bounded regions where the point processes exist, $\Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}} = \int_{S_{\ell-1,k}} \phi_{(\ell,i)\to(\ell-1,k)}(\tau - t)d\tau$, t^x represents an event from the data, and t^z represents an event from the approximate point process. Moreover, $\mathcal{L}$ is the lower bound for the marginal probability density function.

See Appendix E.1 for the proof.

5.4.1 Approximate the Intensity Function

The ELBO in Equation 5.4 is a functional *w.r.t* $\{\lambda^E_{\ell,i}(\cdot)\}$ and the direct way to minimize the functional is to calculate the functional derivative and find a critical point for the functional. But the expression for the ELBO is quite complicated and there is no simple expression for the critical point.

For simplicity and demonstration purpose, we only consider approximating DNSPs for TPPs as in Example 2.4. So we parameterize the *cumulative intensity function* using the rational quadratic splines as by Gregory and Delbourgo (1982); Durkan et al. (2019); Shchur et al. (2020b), and the intensity function is just the derivative of the cumulative intensity function. As mentioned in Section 3.2, the intensity function modeled using rational quadratic splines is very flexible and sampling from the IPP is extremely fast.

Parameterization of the spline

We follow the notations and expressions as of Durkan et al. (2019). The spline g maps $[a, b]$ to $[g_a, g_b]$ and a set of knots $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$ is selected *s.t.* they satisfy the following conditions:

$$(x^{(0)}, y^{(0)}) = (a, g_a), \tag{5.5}$$

$$(x^{(K)}, y^{(K)}) = (b, g_b), \tag{5.6}$$

$$x^{(k)} < x^{(k+1)} \text{ and } y^{(k)} < y^{(k+1)} \text{ for all } k = 0, \cdots, K-1. \tag{5.7}$$

We use $\{\delta^{(k)}\}_{k=0}^{K}$ to denote the non-negative derivatives of the spline at the knots. The monotonic rational quadratic spline given by Gregory and Delbourgo (1982) passes through every knot. The functional form for the spline is

$$g(x) = \frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)},\ x \in [x^{(k)}, x^{(k+1)}]$$

where

$$\alpha^{(k)}(\xi) = s^{(k)} y^{(k+1)} \xi^2 + \left[y^{(k)} \delta^{(k+1)} + y^{(k+1)} \delta^{(k)} \right] \xi(1-\xi) + s^{(k)} y^{(k)} (1-\xi)^2, \tag{5.8}$$

$$\beta^{(k)}(\xi) = s^{(k)} + \left[\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)} \right] \xi(1-\xi), \tag{5.9}$$

$$s^{(k)} = \frac{(y^{(k+1)} - y^{(k)})}{w^{(k)}}, \tag{5.10}$$

$$\xi = \frac{x - x^{(k)}}{w^{(k)}}, \tag{5.11}$$

$$w^{(k)} = x^{(k+1)} - x^{(k)}. \tag{5.12}$$

The details of the inverse and derivative for g can be found in Durkan et al. (2019).

Parameterization of the transformation

Suppose we want to generate samples for a IPP for an interval $[0, T]$, then

$$\Lambda(t) = \lambda \cdot g\left(t \cdot \frac{1}{T}\right),$$

where g is a rational quadratic spline mapping from $(0,1)$ to $(0,1)$, and λ is a positive parameter.

5.4.2 Synthetic Results

We use a synthetic experiment to demonstrate that our approximate posterior point process is very close to the true posterior point process. The structures we use here for DNSPs are 1-hidden and 2-hidden models. The number of types for both of them is 1.

Kernel We choose a gamma kernel as in Example 2.4. Let $p = 1$, $\alpha = 1$ and $\beta = 1$, then the kernel function is simply

$$\phi_{\boldsymbol{\theta}}(x) = \begin{cases} e^{-x}, \text{ for } x > 0, \\ 0, \text{ for } x \leq 0. \end{cases}$$

Differential Relaxation There is a discontinuous point for the kernel function at 0 and the ELBO is not differentiable *w.r.t* the parameters of the spline. Like the relaxation trick in Shchur et al. (2020b), we make the kernel differentiable by multiplying the kernel function by a sigmoid function and the new kernel function becomes

$$\phi_{\boldsymbol{\theta}}(x) = e^{-x} \cdot \frac{1}{1 + e^{-x/\gamma}}\ ,\ \gamma \in (0,1).$$

The smaller the γ is, the more accurate the approximation is. But a small value for γ makes training harder.

Synthetic Data We assume there is only one type of event and the time interval is $[0, 20]$. The events for the evidence are at times at 6 and 10.

Intensity We compare the intensity function inferenced from rational quadratic spline and the intensity estimated using MCMC samples. The intensity function produced by VI is simply $\frac{\partial}{\partial t}\Lambda(t)$. For the estimation from MCMC samples, we first divide the whole time interval into small sub-intervals. For each small sub-interval, we count the number of events falling into that sub-interval and then divide the number by the length of that small sub-interval to get the intensity for that small sub-interval.

The comparison results are given in Figure 5.3 and Figure 5.4, where the blue circles are knots. It is obvious from the figures that the intensity function given by VI closely follows the intensity function estimated by using MCMC samples.

5.5 Variational Inference with Inclusive KL Divergence

The mean-field approximation does not include any prior information from the observation, which makes it hard to train and we cannot directly get the approximation based on the observation. Hence, we design another class of approximate posterior point processes with hierarchical structures, and the shapes of the approximate posterior point processes depend on the observed data. Surprisingly, if we let the approximate posterior point processes be our virtual point processes, the approximation can be learned using Algorithm 1. During the training process, lines 4 and 5 in Algorithm 1 are used to maximize the marginal likelihood, line 6 in Algorithm 1 is used to minimize the inclusive KL divergence. The

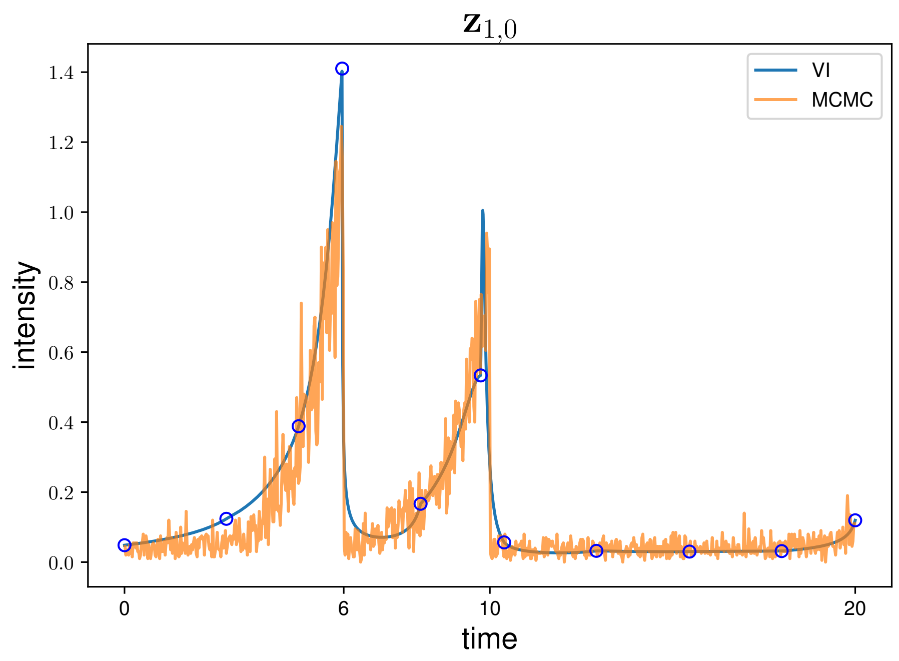


Figure 5.3: Comparison of the intensity function between VI and MCMC for 1-hidden model.

parameters for the virtual point processes can serve as our parameters for the approximate point processes. With the help of our approximate point processes, we can avoid the time-consuming mixing steps of MCMC when performing prediction and we are able to achieve better prediction performance for both the time and the type when only a limited amount of time is allowed.

5.5.1 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

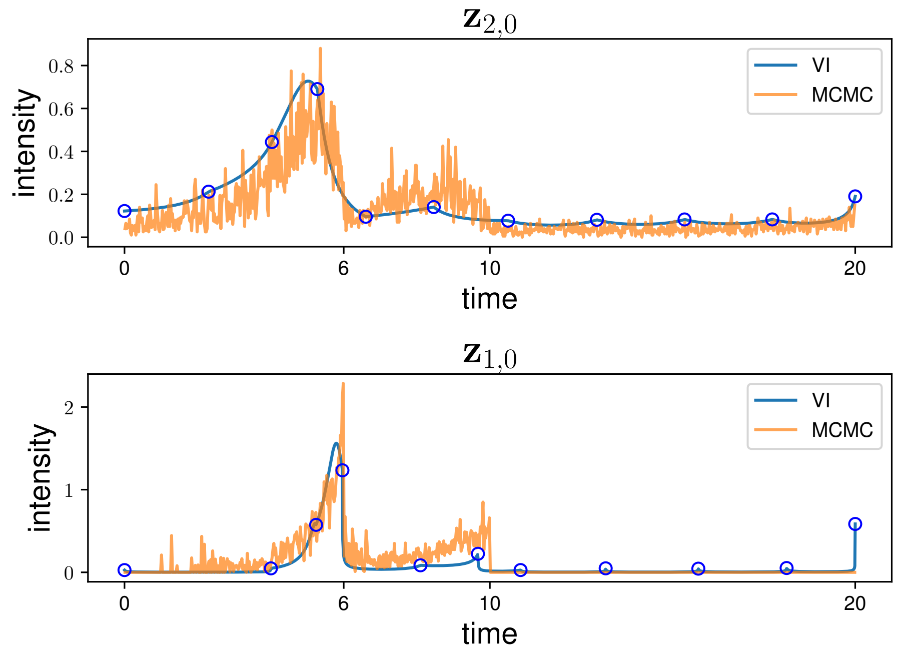


Figure 5.4: Comparison of the intensity function between VI and MCMC for 2-hidden model.

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 $\mathbf{Z}^I = \{\mathbf{Z}^I_1, \cdots, \mathbf{Z}^I_L\}$, with $\mathbf{Z}^I_\ell = \{Z^I_{\ell,k}\}_{k=1}^{K_\ell}$.

$\mathbf{Z}^I_1$ is assumed to be Poisson processes, and the CIF for $Z^I_{1,k}$ is

$$\lambda^I_{1,k}(t) = q^I_{1,k}(t; \mathbf{z}^I_0 = \mathbf{x}, \boldsymbol{\theta}^I_{1,k}(t)), \tag{5.13}$$

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

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

$$\lambda_{\ell,k}^{I}(t) = q_{\ell,k}^{I}(t; \mathbf{z}_{\ell-1}^{I}, \boldsymbol{\theta}_{\ell,k}^{I}(t)), \tag{5.14}$$

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.

5.5.2 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 dissertation: upward Neyman-Scott processes and upward self-attention processes.

Upward Neyman-Scott Processes (UNSP) 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

$$\begin{aligned}\lambda_{\ell,k}^{I}(t) &= q_{\ell,k}^{I}\left(t; \boldsymbol{z}_{\ell-1}^{I}, \boldsymbol{\theta}_{\ell,k}^{I}\right) \\ &= \mu_{\ell,k}^{I} + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}} \phi_{\boldsymbol{\theta}_{(\ell-1,i)\to(\ell,k)}^{I}}(t_{\ell-1,i,j} - t),\end{aligned} \tag{5.15}$$

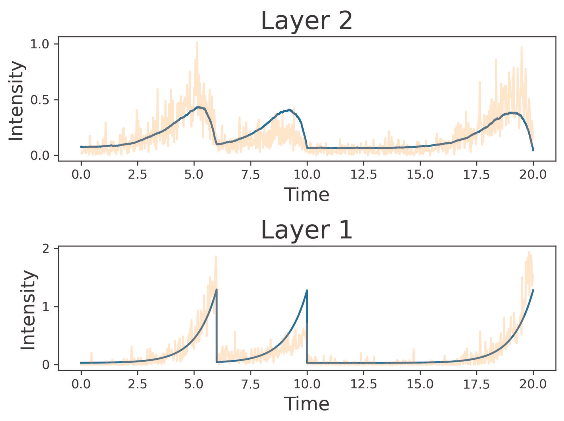
where $\boldsymbol{\Theta}_{\ell,k}^{I} = \left\{\mu_{\ell,k}^{I}, \left\{\boldsymbol{\theta}_{(\ell-1,i)\to(\ell,k)}^{I}\right\}_{i=1}^{K_{\ell-1}}\right\}$ and $\mu_{\ell,k}^{I} \geq 0$. We see that Equation 5.15 has the same functional form as Equation 4.2.

Upward Self-Attention Processes (USAP) 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 $\boldsymbol{h}_{\ell-1,i,j} = f_{\text{Attn}}(t_{\ell-1,i,j}, \mathbf{z}^I_{\ell-1} = \{\{t_{\ell-1,i,j}\}_{j=1}^{m_{\ell-1,i}}\}_{i=1}^{K_{\ell-1}}\})$ through self-attention. The CIF for $t_{\ell-1,i,j-1} < t \leq t_{\ell-1,i,j}$ becomes

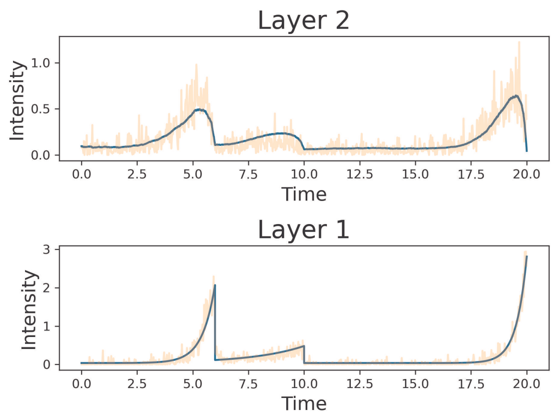
$$\begin{aligned}\lambda^I_{\ell,k}(t) &= q^I_{\ell,k}\left(t; \mathbf{z}^I_{\ell-1}, \boldsymbol{\theta}^I_{\ell,k}\right) \\ &= \mu^I_{\ell,k} + \phi_{\boldsymbol{\theta}^I_{\ell-1,i,j}}(t_{\ell-1,i,j} - t),\end{aligned} \tag{5.16}$$

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

Comparison We compare UNSP, USAP, and MCMC in Figure 5.5. We have 3 events in the observation (times at 6, 10, and 20), and we use MCMC, UNSPs and USAPs to infer 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 $q^I_{1,0}(\cdot)$. For the approximate intensity function for layer 2, we first generate many samples from $q^I_{1,0}(\cdot)$ which induce samples of $q^I_{2,0}(\cdot)$. Then the approximate intensity function for layer 2 is the mean of all the samples for $q^I_{2,0}(\cdot)$. We see that USAP fits the approximate intensity functions estimated by the MCMC better than UNSP.



(a) UNSP vs. MCMC



(b) USAP vs. MCMC

Figure 5.5: Approximate intensity functions estimated by using MCMC and VI.

The better approximation of USAPs in Figure 5.5 comes from the fact that the kernel function can adjust their 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 (Proposition 4.1). Therefore, USAPs are better than UNSPs, since UNSPs only include the information of the events happening after t.

Remark 5.3. *Similar to Remark 4.2, 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.*

5.5.3 Inference

Inference for the Model Parameters

While Section 5.3 view the lines 4 and 5 in Algorithm 1 as a part of ascent-based MCEM, we can also view it as an unbiased estimator of the gradient of the marginal likelihood $\log p(\mathbf{x}; \boldsymbol{\Theta})$ based on the Fisher identity (Ou and Song, 2020; Naesseth et al., 2020),

$$\nabla_{\boldsymbol{\Theta}} \log p(\mathbf{x}; \boldsymbol{\Theta}) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x};\boldsymbol{\Theta})} \left[\nabla_{\boldsymbol{\Theta}} \log p(\mathbf{z}, \mathbf{x}; \boldsymbol{\Theta}) \right]. \tag{5.17}$$

Based on Equation 5.17, 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).

Algorithm 3 Prediction with $\underline{\text{MCMC}}$

Input: observed data $\mathbf{x} = \{e_1, e_2, \cdots, e_n\}$, where $e_i = (t_i, k_i)$, and model $\mathcal{M}$

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

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

```
1: for $s = 1$ to $\mathcal{S}$ do
2:     $\underline{\text{sample } \mathbf{z}^s \sim p(\mathbf{z} \mid \mathbf{x}; \mathbf{\Theta})}$
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:     $\underline{\text{sample } \mathbf{z}^s \sim p(\mathbf{z} \mid \mathbf{x}; \mathbf{\Theta})}$
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: $\hat{k}_{n+1} = \arg\max_{k \in \{1,\ldots,K_0\}} \sum_{s=1}^{\mathcal{S}} \mathbb{1}_k(\hat{k}_{n+1}^s)$
```

Algorithm 4 Prediction with approximation

Input: observed data $\mathbf{x} = \{e_1, e_2, \cdots, e_n\}$, where $e_i = (t_i, k_i)$, and model $\mathcal{M}$

Initialization: $\mathbf{\Theta}$, $\mathbf{\Theta}^I$, and sample size $\mathcal{S}$.

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

```
1: for $s = 1$ to $\mathcal{S}$ do
2:     sample $\mathbf{z}^s \sim q(\mathbf{z}; \mathbf{x}, \mathbf{\Theta}^I)$
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 $\mathbf{z}^s \sim q(\mathbf{z}; \mathbf{x}, \mathbf{\Theta}^I)$
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: $\hat{k}_{n+1} = \arg\max_{k \in \{1,\ldots,K_0\}} \sum_{s=1}^{\mathcal{S}} \mathbb{1}_k(\hat{k}_{n+1}^s)$
```

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 $D_{KL}(p \parallel q^I)$ is

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

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

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

The gradient of the KL divergence $w.r.t$ the variational parameters $\boldsymbol{\Theta}^I$ is

$$\mathbb{E}_{p(\mathbf{z}|\mathbf{x};\boldsymbol{\Theta})}[-\nabla_{\boldsymbol{\Theta}^I} \log q(\mathbf{z}; \mathbf{x}, \boldsymbol{\Theta}^I)], \tag{5.18}$$

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

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 Eqs. 5.18, 4.4, and 4.5, 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 $\tilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta}^I$).

Therefore, we can use Algorithm 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 Section 5.5.4.

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

5.5.4 Prediction

We rewrite the prediction procedures with MCMC in Algorithm 3. Suppose we are given a sequence of events $\{e_1, e_2, \cdots, e_n\}$, 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 $t_1 \leq t_2 \leq t_3 \leq \cdots \leq t_n$. We initialize RPPs with parameters $\boldsymbol{\Theta}$, VPPs with parameters $\tilde{\boldsymbol{\Theta}}$. These parameters were obtained by MCEM run 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 $m^s_{L,k}$ and the time period is $[0, T]$, then the MLE for the constant rate is $\mu_k = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} \frac{m^s_{L,k}}{T}$. 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 $(t_{n+1}^1, k_{n+1}^1), (t_{n+1}^2, k_{n+1}^2), \ldots, (t_{n+1}^{\mathcal{S}}, k_{n+1}^{\mathcal{S}})$, 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 $\hat{t}_{n+1} = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} t_{n+1}^s$ and the type prediction is $\hat{k}_{n+1} = \arg\max_{k \in \{1,\ldots,K_0\}} \sum_{s=1}^{\mathcal{S}} \mathbb{1}_k(k_{n+1}^s)$, where $\mathbb{1}_k(k_{n+1}^s)$ is the indicator function which 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.

Algorithm 4 is our proposed method for prediction with our approximate posterior point processes. We replace the MCMC sampling with the sampling from our approximate posterior point processes. Sampling from $q(\mathbf{z}; \mathbf{x}, \mathbf{\Theta}^I)$ 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 Section 5.5.5.

5.5.5 Experiments

Like Section 5.3.1, 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 Section 5.3.1. 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.

In Figures 5.6, 5.7, 5.8 and 5.9, we use different line styles with different colors to represent the results from different models or algorithms. 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.

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 USAPs in terms of time prediction, and the type predictions of USAPs and UNSPs are similar. Moreover, DNSPs-based methods are better than non-DNSPs-based methods for all these real-world datasets.

Datasets

Synthetic Datasets The point processes are defined in the interval $(0, 20]$. The constant rate for the top layer is a fixed number across different sequences. 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.

The retweet, earthquake, homicide datasets are the same as in Section 5.3.1.

The statistics of the datasets are shown in Table 5.5.

Table 5.5: Datasets Statistics for Predictions

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

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 5.6, where d_k, d_v, d_M, and d_H appear in Eqs. F.2 and F.3, MPSS stands for model parameters step size, and SASS stands for self-attention step size.

Table 5.6: 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

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.

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.

5.5.6 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 1-hidden DNSP to learn from the dataset generated from 1-hidden, and 2-hidden DNSP to from learn 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.

Figure 5.6 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 (Figures 5.6a and 5.6c). 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 (Figure 5.6d) and they have similar performance for the 1-hidden synthetic dataset (Figure 5.6b).

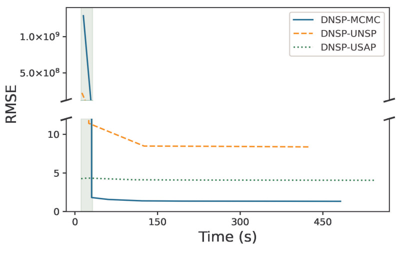
5.5.7 Real-World Data Experiments

The datasets we use are of retweets (Zhao et al., 2015), earthquakes (NCEDC, 2014; BDSN, 2014; HRSN, 2014; BARD, 2014), and homicides (COC) . 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 Section 5.3.1. We do not fix k for the Weibull kernel as we did in the synthetic data experiments.

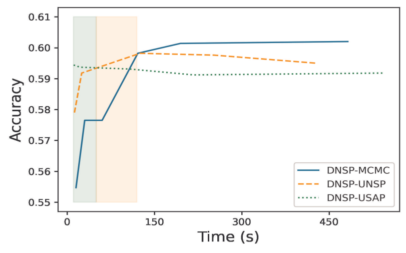
Figures 5.7, 5.8 and 5.9 summarize the real-world experimental results for retweets, earthquakes and homicides respectively. Similar to the experimental results for the synthetic dataset, UNSPs and USAPs are better than MCMC for both the time prediction and the type prediction when we only have a small 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 (Figures 5.7a, 5.7c, 5.8a, 5.8c, 5.9a, and 5.9c). UNSPs or USAPs are better than other non-DNSPs baselines for all these experiments (Figures 5.7a, 5.7c, 5.8a, 5.8c, 5.9a, and 5.9c). USAPs are better than UNSPs for all of these experiments in the end (Figures 5.7a, 5.7c, 5.8a, 5.8c, 5.9a, and 5.9c). UNSPs are better than USAPs for retweets and earthquakes when we only have a small number of samples (Figures 5.7a, 5.7c, 5.8a, and 5.8c). USAPs are always better than UNSPs for homicides in terms of time prediction (Figures 5.9a and 5.9c).

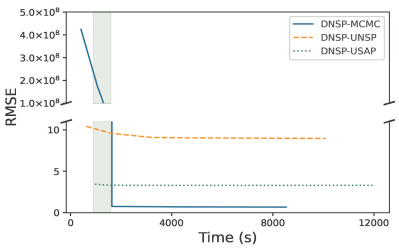
For the type prediction, DNSPs with MCMC or approximate posterior point processes achieve the best prediction results for all these datasets (Figure 5.7d for retweets, 5.8d for earthquakes, Figure 5.9d for homicides). UNSPs or USAPs are better than other non-DNSPs baselines for all these datasets (Figure 5.7d for retweets, 5.8d for earthquakes, Figure 5.9d for homicides). In Figures 5.7b and 5.9d, USAPs are better than UNSPs, while UNSPs are better than USAPs in Figures 5.7d and 5.8d. USAPs and UNSPs have similar performance in Figures 5.8b and 5.9b.



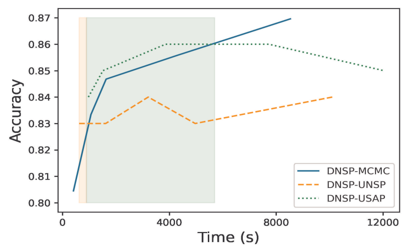
(a) 1-hidden RMSE



(b) 1-hidden accuracy

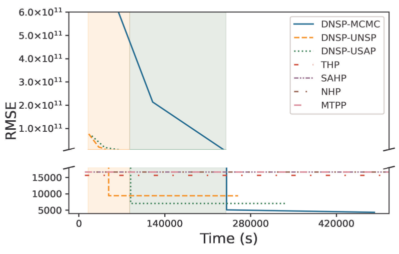


(c) 2-hidden RMSE

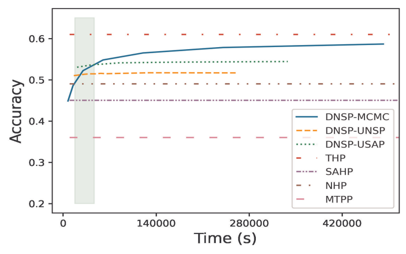


(d) 2-hidden accuracy

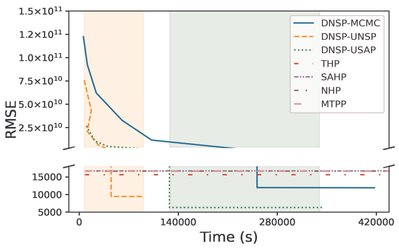
Figure 5.6: Synthetic Dataset



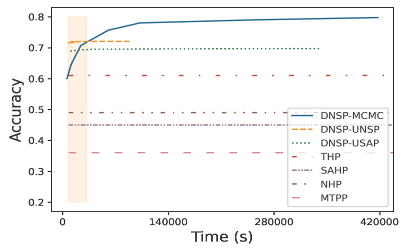
(a) 1-hidden RMSE



(b) 1-hidden accuracy

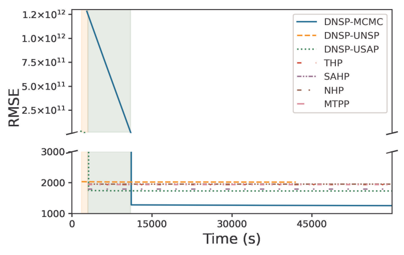


(c) 2-hidden RMSE

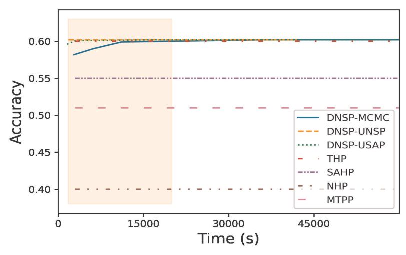


(d) 2-hidden accuracy

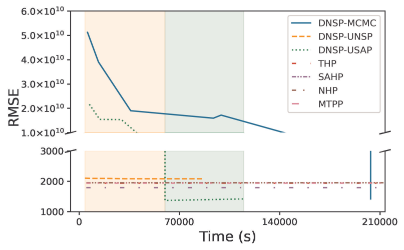
Figure 5.7: Retweet Dataset



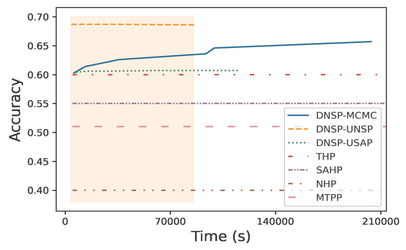
(a) 1-hidden RMSE



(b) 1-hidden accuracy

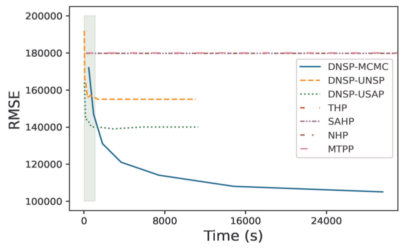


(c) 2-hidden RMSE

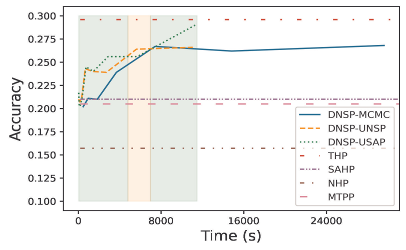


(d) 2-hidden accuracy

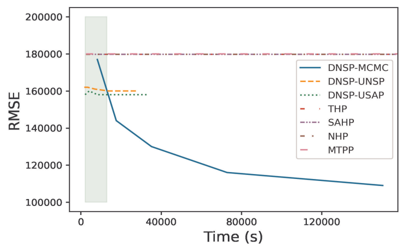
Figure 5.8: Earthquake Dataset



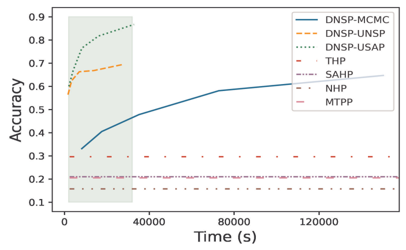
(a) 1-hidden RMSE



(b) 1-hidden accuracy



(c) 2-hidden RMSE



(d) 2-hidden accuracy

Figure 5.9: Homicide Dataset

Chapter 6

Conclusion

We build a deep Neyman-Scott process (DNSP) and use it to model real-world event sequences. Different from the existing methods for Cox processes, we are able to stack point processes in a hierarchical manner and do not assume the intensity function is smooth in the space. We propose and test an efficient MCMC posterior sampling algorithm for DNSPs. Virtual events as auxiliary variables help accelerate the mixing of our Markov chains. With the fast posterior sampling, we are able to do inference for large datasets. Our encouraging experiments results suggest that it is promising to build a deep model with point processes.

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

Bibliography

Ryan Prescott Adams. *Kernel methods for nonparametric Bayesian inference of probability densities and point processes.* PhD thesis, University of Cambridge, 2009.

Ryan Prescott Adams, Iain Murray, and David JC MacKay. Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16, 2009.

Virginia Aglietti, Edwin V Bonilla, Theodoros Damoulas, and Sally Cripps. Structured variational inference in continuous Cox process models. In *Advances in Neural Information Processing Systems*, pages 12458–12468, 2019.

Ina Trolle Andersen, Ute Hahn, Eva C Arnspang, Lene Niemann Nejsum, and Eva B Vedel Jensen. Double Cox cluster processes—with applications to photoactivated localization microscopy. *Spatial statistics*, 27:58–73, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Adrian Baddeley and Jesper Møller. Nearest-neighbour Markov point processes and random sets. *International Statistical Review/Revue Internationale de Statistique*, pages 89–121, 1989.

BARD. Bay area regional deformation network. UC Berkeley Seismological Laboratory. dataset., 2014.

Luc Bauwens and Nikolaus Hautsch. Modelling financial high frequency data using point processes. In *Handbook of financial time series*, pages 953–979. Springer, 2009.

BDSN. Berkeley digital seismic network. UC Berkeley Seismological Laboratory. dataset., 2014.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Brian S Caffo, Wolfgang Jank, and Galin L Jones. Ascent-based Monte Carlo expectation-maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):235–251, 2005.

Erhan Çinlar. *Introduction to stochastic processes.* Dover Publications, 2013.

Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XQQA6-So14.

Peter Clifford and Geoff Nicholls. Comparison of birth-and-death and Metropolis-Hastings Markov chain Monte Carlo for the Strauss process. Technical report, 1994.

COC. City of Chicago, Crimes - 2001 to present. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2.

David R Cox. Some statistical methods connected with series of events. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):129–157, 1955.

D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer-Verlag New York, 2nd edition, 2003.

Christian Donner and Manfred Opper. Efficient Bayesian inference of sigmoidal Gaussian Cox processes. *The Journal of Machine Learning Research*, 19(1):2710–2743, 2018.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564, 2016.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7511–7522. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

Charles J Geyer and Jesper Møller. Simulation procedures and likelihood inference for spatial point processes. *Scandinavian journal of statistics*, pages 359–373, 1994.

Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

Peter J Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

JA Gregory and R Delbourgo. Piecewise rational quadratic interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 2(2):123–130, 1982.

Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, pages 1962–1970, 2011.

Tom Gunter, Chris M. Lloyd, Michael A. Osborne, and Stephen J. Roberts. Efficient Bayesian nonparametric modelling of structured point processes. In Nevin L. Zhang and Jin Tian, editors, *Proceedings of the Thirtieth Conference on Uncertainty in Artificial*

Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014, pages 310–319. AUAI Press, 2014.

W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

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

HRSN. High resolution seismic network. UC Berkeley Seismological Laboratory. dataset., 2014.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Olav Kallenberg. An informal guide to the theory of conditioning in point processes. *International Statistical Review/Revue Internationale de Statistique*, pages 151–164, 1984.

Frank P Kelly and Brian D Ripley. A note on Strauss's model for clustering. *Biometrika*, pages 357–360, 1976.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Solomon Kullback. *Information theory and statistics.* Courier Corporation, 1997.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Wenzhao Lian, Ricardo Henao, Vinayak Rao, Joseph Lucas, and Lawrence Carin. A multitask point process predictive model. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2030–2038, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/lian15.html.

Van Lieshout, M. N. M., and A. J. Baddeley. Extrapolating and interpolating spatial patterns. In Andrew B Lawson and David GT Denison, editors, *Spatial Cluster Modelling*, chapter 4, pages 61–86. Chapman & Hall/CRC, 2002.

Chris Lloyd, Tom Gunter, Michael Osborne, and Stephen Roberts. Variational inference for Gaussian process modulated Poisson processes. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1814–1822, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/lloyd15.html.

B Matérn. Spatial variation: Meddelanden fran statens skogsforskningsinstitut. *Lecture Notes in Statistics*, 36:21, 1960.

Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3165–3174, 2019.

Hongyuan Mei and Jason M Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, pages 6754–6764, 2017.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Jesper Møller. On the rate of convergence of spatial birth-and-death processes. *Annals of the Institute of Statistical Mathematics*, 41(3):565–581, 1989.

Jesper Møller and Rasmus Plenge Waagepetersen. *Statistical inference and simulation for spatial point processes*. CRC Press, 2003.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log Gaussian Cox processes. *Scandinavian journal of statistics*, 25(3):451–482, 1998.

Christian Naesseth, Fredrik Lindsten, and David Blei. Markovian score climbing: Variational inference with KL(p || q). In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15499–15510. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/b20706935de35bbe643733f856d9e5d6-Paper.pdf.

NCEDC. Northern California earthquake data center. UC Berkeley Seismological Laboratory. dataset., 2014.

Ronald C Neath et al. 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, 2013.

Jerzy Neyman and Elizabeth L Scott. Statistical approach to problems of cosmology. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(1):1–29, 1958.

GE Norman and VS Filinov. Investigations of phase transitions by a Monte-Carlo method. *High Temperature*, 7(2):216, 1969.

Takahiro Omi, Kazuyuki Aihara, et al. Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems*, pages 2122–2132, 2019.

Zhijian Ou and Yunfu Song. Joint stochastic approximation and its application to learning discrete latent variable models. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 929–938. PMLR, 03–06 Aug 2020. URL https://proceedings.mlr.press/v124/ou20a.html.

Fredos Papangelou. The conditional intensity of general point processes and an application to line processes. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 28(3): 207–226, 1974.

Donald H Perkel, George L Gerstein, and George P Moore. Neuronal spike trains and stochastic point processes: I. the single spike train. *Biophysical journal*, 7(4):391–418, 1967.

Chris Preston. Spatial birth and death processes. *Bulletin of the International Statistical Institute*, 46:371–391, 1977.

Zhen Qin and Christian R. Shelton. Auxiliary gibbs sampling for inference in piecewise-constant conditional intensity models. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, page 722–731, Arlington, Virginia, USA, 2015. AUAI Press. ISBN 9780996643108.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, 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, 22–25 Apr 2014. PMLR.

Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 619–626, 2011.

Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and extensions. *The Journal of Machine Learning Research*, 14(1):3295–3320, 2013.

Brian D Ripley. Modelling spatial patterns. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):172–192, 1977.

Oleksandr Shchur, Marin Bilos, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=HygOjhEYDH.

Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. *Advances in Neural Information Processing Systems*, 33, 2020b.

Christian Shelton, Zhen Qin, and Chandini Shetty. Hawkes process inference with missing data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.12116. URL https://ojs.aaai.org/index.php/AAAI/article/view/12116.

R.S. Stoica, E. Tempel, L.J. Liivamägi, G. Castellan, and E. Saar. Spatial patterns analysis in cosmology based on marked point processes. *European Astronomical Society Publications Series*, 66:197–226, 2014. doi: 10.1051/eas/1466013.

Marjorie Thomas. A generalization of Poisson's binomial limit for use in ecology. *Biometrika*, 36(1/2):18–25, 1949.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Greg CG Wei and Martin A Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411):699–704, 1990.

Jeremy C Weiss and David Page. Forest-based point process for event prediction from electronic health records. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2013.

Chun Yip Yau and Ji Meng Loh. A generalization of the Neyman-Scott process. *Statistica Sinica*, pages 1717–1736, 2012.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive Hawkes process. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11183–11193. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/zhang20q.html.

Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 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, 2015.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes process. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11692–11702. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/zuo20a.html.

Appendix A

Posterior PCIF

Lemma A.1 (Kallenberg (1984)). *The PCIF for a point process Ξ defined on S with density f is*

$$\lambda_{\mathcal{P}}(t) = \frac{f(\xi \cup \{t\})}{f(\xi)},\ t \in S \backslash \xi,$$

with ξ as a realization of Ξ and $a/0 = 0$ for $a \geq 0$.

A.1 Proof of Proposition 4.1

Proposition 4.1. *For temporal DNSPs, the Papangelou conditional intensity function for the posterior point process of $Z_{\ell,k}$ is*

$$\begin{aligned}\lambda_{\mathcal{P};\ell,k}(t) =& \lambda_{\ell,k}(t) \\ & \cdot \left(\prod_{i=1}^{K_{\ell-1}} \left(\exp\left(-\Phi_{(\ell,k)\to(\ell-1,i)}(T-t)\right) \prod_{t_{\ell-1,i,j}>t} \frac{\lambda'_{\ell-1,i}(t_{\ell-1,i,j},t)}{\lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right) \right),\end{aligned} \tag{4.1}$$

where

$$\lambda'_{\ell-1,i}(x,t) = \lambda_{\ell-1,i}(x) + \phi_{(\ell,k)\to(\ell-1,i)}(x-t)$$

and

$$\Phi_{(\ell,k)\to(\ell-1,i)}(x) = \int_0^x \phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(\tau)d\tau.$$

Proof. The $p.d.f$ to have an event at time t for $Z_{\ell,k}$ is

$$p(t, \mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1}, \mathbf{z}_\ell/\{t\}, \mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_L) = p(\mathbf{z}_L)p(\mathbf{z}_{L-1} \mid \mathbf{z}_L)\cdots p(\mathbf{x} \mid \mathbf{z}_1),$$

where $\mathbf{z}_\ell/\{t\}$ means there is no event at time t for $Z_{\ell,k}$.

If there is no event at time t, the $p.d.f$ conditional on the information of the point processes except time t is

$$p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1}, \mathbf{z}_\ell/\{t\}, \mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_L) = p(\mathbf{z}_L)p(\mathbf{z}_{L-1} \mid \mathbf{z}_L)\cdots p(\mathbf{z}_\ell/\{t\} \mid \mathbf{z}_{\ell+1})\cdots p(\mathbf{x} \mid \mathbf{z}_1).$$

Thus, according to Lemma A.1, the posterior PCIF to have an event at time t is

$$\begin{aligned}
\lambda_{\mathcal{P};\ell,k}(t) &= \frac{p(t, \mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1}, \mathbf{z}_\ell/\{t\}, \mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_L)}{p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1}, \mathbf{z}_\ell/\{t\}, \mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_L)} \\
&= \frac{p(\mathbf{z}_\ell/\{t\} \cup \{t\} \mid \mathbf{z}_{\ell+1})}{p(\mathbf{z}_\ell/\{t\} \mid \mathbf{z}_{\ell+1})} \\
&= \lambda_{\ell,k}(t) \prod_{i=1}^{K_{\ell-1}} \left(\exp\left(-\Phi_{(\ell,k)\to(\ell-1,i)}(T-t)\right) \prod_{t_{\ell-1,i,j}>t} \frac{\lambda'_{\ell-1,i}(t_{\ell-1,i,j}, t)}{\lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right).
\end{aligned}$$

□

Appendix B

Spatial Birth-and-Death Algorithm

We attempt to construct a spatial birth-death process to simulate the posterior distribution of hidden point processes. We need to construct two proposals: birth and death. We use a birth to add an event and a death to delete an event.

Spatial birth-and-death Preston (1977) is a continuous-time Markov process. The detailed balance equation is

$$p(\mathbf{z}_{\ell,k} \mid \mathbf{x})b(\mathbf{z}_{\ell,k}, t) = p(\mathbf{z}_{\ell,k} \cup \{t\} \mid \mathbf{x})d(\mathbf{z}_{\ell,k} \cup \{t\}, t), \tag{B.1}$$

where $b(\mathbf{z}_{\ell,k}, t)$ is the birth rate to add a new event at time t to the current hidden events set $\mathbf{z}_{\ell,k}$ and $d(\mathbf{z}_{\ell,k}, t)$ is the death rate for removing an event with time t.

A common way to determine the birth rate $b(\mathbf{z}_{\ell,k}, t)$ is to make it proportional to the PCIF as in Equation 4.1 and the death rate to be a constant number Ripley (1977); Baddeley and Møller (1989); Møller (1989). However, it would be very hard to calculate the total birth rate exactly, as it would require an integral of the product terms in Equation 4.1. If, instead, we try to find an upper bound for the PCIF and then use thinning to get the samples for the birth process, it would have far too many attempted jumps rejected for the birth stage.

Similar to Lieshout et al. (2002), we can make the birth rate to be

$$b(\mathbf{z}_{\ell,k}, t) = \tilde{\mu}_{\ell,k} + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}>t} \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j} - t). \tag{B.2}$$

The only difference from Lieshout et al. (2002) is we divide the birth rate by $\lambda_{\ell,k}(\cdot)$ and the birth rate not only depends on the evidence, but also depends on the posterior samples.

To satisfy the detailed balance Equation B.1, the death rate $d(\mathbf{z}_{\ell,k} \cup t, t)$ needs to be

$$\begin{aligned} d(\mathbf{z}_{\ell,k} \cup t, t) &= \frac{b(\mathbf{z}_{\ell,k}, t)}{\lambda_{\mathcal{P};\ell,k}(t)} \\ &= \frac{\tilde{\mu}_{\ell,k} + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}>t} \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j} - t)}{\lambda_{\ell,k}(t) \prod_{i=1}^{K_{\ell-1}} \left(\exp\left(-\Phi_{(\ell,k)\to(\ell-1,i)}(T-t)\right) \prod_{t_{\ell-1,i,j}>t} \frac{\lambda'_{\ell-1,i}(t_{\ell-1,i,j},t)}{\lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right)}, \end{aligned} \tag{B.3}$$

where $\lambda'_{\ell-1,i}(x,y) = \lambda_{\ell-1,i}(x) + \phi_{\theta_{(\ell,k)\to(\ell-1,i)}}(x-y)$.

The total birth rate is

$$\beta(\mathbf{z}) = \sum_{\ell,k} \int_0^T b(\mathbf{z}_{\ell,k}, t)dt = \sum_{\ell,k} \left(\tilde{\mu}_{\ell,k} T + \sum_{i=1}^{K_{\ell-1}} \sum_{t_{\ell-1,i,j}} \tilde{\Phi}_{\tilde{\theta}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j}) \right), \tag{B.4}$$

and the total death rate is

$$\delta(\mathbf{z}) = \sum_{\ell,k} \left(\sum_j d(\mathbf{z}_{\ell,k}, t_{\ell,k,j}) \right).$$

Open problem: *Does the SB&D with the birth rate as in Equation B.2 and the death rate as in Equation B.3 converge to an invariant distribution?*

The SB&D given in Lieshout et al. (2002) is guaranteed to converge to an invariant distribution as it satisfies a sufficient condition that the total birth rate is bounded from above and the total death rate is bounded from below. However, the total birth rate in Equation B.4 is finite but not bounded in our case, which violates the sufficient condition satisfied in Lieshout et al. (2002). Please refer to Møller and Waagepetersen (2003) for more details.

Appendix C

Acceptance Probability

We follow the notation in Section 4.1.

Recall

$$\mathcal{P} = \frac{p(\mathbf{x}, \mathbf{z}', \tilde{\mathbf{z}}')}{p(\mathbf{x}, \mathbf{z}, \tilde{\mathbf{z}})} = \frac{p(z'_{\ell,k} \mid \mathbf{z}_{\ell+1}) \cdot \tilde{p}(\tilde{z}'_{\ell,k} \mid \mathbf{z}_{\ell-1})}{p(z_{\ell,k} \mid \mathbf{z}_{\ell+1}) \cdot \tilde{p}(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1})} \cdot \frac{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}'_{\ell}) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}'_{\ell})}{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}_{\ell}) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_{\ell})},$$

where

$$\begin{aligned} p(\mathbf{z}_{\ell-1} \mid \mathbf{z}_{\ell}) &= \prod_{k=1}^{K_{\ell-1}} p(z_{\ell-1,k} \mid \mathbf{z}_{\ell}), \\ \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_{\ell}) &= \prod_{k=1}^{K_{\ell+1}} \tilde{p}(\tilde{z}_{\ell+1,k} \mid \mathbf{z}_{\ell}). \end{aligned}$$

C.1 Re-sample Virtual Events

Suppose we want to re-sample the virtual events for $\tilde{Z}_{\ell,k}$, the proposal probability is

$$q(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1}) = \tilde{p}(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1}),$$

and the integrated detailed balance equation is

$$\begin{aligned}&\int_A \int_{A'} p(\mathbf{x}, \mathbf{z}, \tilde{\mathbf{z}}) q(\tilde{z}'_{\ell,k} \mid \mathbf{z}_{\ell-1}) \alpha(\tilde{z}_{\ell,k}, \tilde{z}'_{\ell,k}) d\tilde{z}'_{\ell,k} d\tilde{z}_{\ell,k} \\ =& \int_A \int_{A'} p(\mathbf{x}, \mathbf{z}', \tilde{\mathbf{z}}') q(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1}) \alpha(\tilde{z}'_{\ell,k}, \tilde{z}_{\ell,k}) d\tilde{z}'_{\ell,k} d\tilde{z}_{\ell,k},\end{aligned}$$

where $\tilde{z}_{\ell,k} \in A$ and $\tilde{z}'_{\ell,k} \in A'$.

The proposal probability ratio is

$$\mathcal{Q} = \frac{q(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1})}{q(\tilde{z}'_{\ell,k} \mid \mathbf{z}_{\ell-1})} = \frac{\tilde{p}(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1})}{\tilde{p}(\tilde{z}'_{\ell,k} \mid \mathbf{z}_{\ell-1})}.$$

The likelihood ratio is

$$\mathcal{P} = \frac{p(\mathbf{x}, \mathbf{z}', \tilde{\mathbf{z}}')}{p(\mathbf{x}, \mathbf{z}, \tilde{\mathbf{z}})} = \frac{\tilde{p}(\tilde{z}'_{\ell,k} \mid \mathbf{z}_{\ell-1})}{\tilde{p}(\tilde{z}_{\ell,k} \mid \mathbf{z}_{\ell-1})}.$$

The absolute value of the determinant of the Jacobian is

$$\mathcal{J} = \left| \frac{\partial(\tilde{z}'_{\ell,k}, \tilde{z}_{\ell,k})}{\partial(\tilde{z}'_{\ell,k}, \tilde{z}_{\ell,k})} \right| = 1.$$

So the acceptance probability is

$$\alpha(\tilde{z}_{\ell,k}, \tilde{z}'_{\ell,k}) = \mathcal{P} \cdot \mathcal{Q} \cdot \mathcal{J} = 1.$$

C.2 Flip a Virtual Event to a Real Event

Suppose we want to flip a virtual event $\tilde{t}_{\ell,k,j}$ to a real event $t_{\ell,k,m_{\ell,k}+1}$.

The number of events for the current state and the proposed state is $m_{\ell,k} + \tilde{m}_{\ell,k}$. The probabilities to pick $\tilde{t}_{\ell,k,j}$ and $t_{\ell,k,m_{\ell,k}+1}$ are both $1/(m_{\ell,k} + \tilde{m}_{\ell,k})$, as the total number of events does not change. Hence, the proposal probability ratio is

$$\mathcal{Q} = \frac{1/(m_{\ell,k} + \tilde{m}_{\ell,k})}{1/(m_{\ell,k} + \tilde{m}_{\ell,k})} = 1.$$

The likelihood ratio is

$$\begin{aligned}
\mathcal{P}_v =& \frac{\lambda_{\ell,k}(\tilde{t}_{\ell,k,j})}{\tilde{\lambda}_{\ell,k}(\tilde{t}_{\ell,k,j})} \cdot \frac{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}'_\ell) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}'_\ell)}{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}_\ell) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_\ell)} \\
=& \frac{\lambda_{\ell,k}(\tilde{t}_{\ell,k,j})}{\tilde{\lambda}_{\ell,k}(\tilde{t}_{\ell,k,j})} \\
& \cdot \prod_{i=1}^{K_{\ell-1}} \left(\exp\left(-\Phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(T - \tilde{t}_{\ell,k,j})\right) \frac{\prod_{t_{\ell-1,i,j} > \tilde{t}_{\ell,k,j}} \lambda'_{\ell-1,i}(t_{\ell-1,i,j}, \tilde{t}_{\ell,k,j})}{\prod_{t_{\ell-1,i,j} > \tilde{t}_{\ell,k,j}} \lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right) \\
& \cdot \prod_{i=1}^{K_{\ell+1}} \left(\exp\left(-\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(\tilde{t}_{\ell,k,j})\right) \frac{\prod_{\tilde{t}_{\ell+1,i,j} < \tilde{t}_{\ell,k,j}} \tilde{\lambda}'_{\ell+1,i}(\tilde{t}_{\ell+1,i,j}, \tilde{t}_{\ell,k,j})}{\prod_{\tilde{t}_{\ell+1,i,j} < \tilde{t}_{\ell,k,j}} \tilde{\lambda}_{\ell+1,i}(\tilde{t}_{\ell+1,i,j})} \right),
\end{aligned}$$

where $\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(x) = \int_0^x \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(\tau)d\tau$, $\lambda'_{\ell-1,i}(x,y) = \lambda_{\ell-1,i}(x) + \phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(x-y)$, and $\tilde{\lambda}'_{\ell+1,i}(x,y) = \tilde{\lambda}_{\ell+1,i}(x) + \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(y-x)$.

So the acceptance probability is

$$\alpha = \min(1, \mathcal{P}_v).$$

C.3 Flip a Real Event to a Virtual Event

Suppose we want to flip a real event $t_{\ell,k,j}$ to a virtual event $\tilde{t}_{\ell,k,\tilde{m}_{\ell,k}+1}$.

Similar to Section C.2, the proposal probability ratio is $\mathcal{Q} = 1$.

The likelihood ratio is

$$\begin{aligned}
\mathcal{P}_r =& \frac{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})}{\lambda_{\ell,k}(t_{\ell,k,j})} \cdot \frac{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}'_\ell) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}'_\ell)}{p(\mathbf{z}_{\ell-1} \mid \mathbf{z}_\ell) \cdot \tilde{p}(\tilde{\mathbf{z}}_{\ell+1} \mid \mathbf{z}_\ell)} \\
=& \frac{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})}{\lambda_{\ell,k}(t_{\ell,k,j})} \\
& \cdot \prod_{i=1}^{K_{\ell-1}} \left(\exp\left(\Phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(T - t_{\ell,k,j})\right) \frac{\prod_{t_{\ell-1,i,j} > t_{\ell,k,j}} \lambda'_{\ell-1,i}(t_{\ell-1,i,j}, t_{\ell,k,j})}{\prod_{t_{\ell-1,i,j} > t_{\ell,k,j}} \lambda_{\ell-1,i}(t_{\ell-1,i,j})} \right) \\
& \cdot \prod_{i=1}^{K_{\ell+1}} \left(\exp\left(\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(t_{\ell,k,j})\right) \frac{\prod_{\tilde{t}_{\ell+1,i,j} < t_{\ell,k,j}} \tilde{\lambda}'_{\ell+1,i}(\tilde{t}_{\ell+1,i,j}, t_{\ell,k,j})}{\prod_{\tilde{t}_{\ell+1,i,j} < t_{\ell,k,j}} \tilde{\lambda}_{\ell+1,i}(\tilde{t}_{\ell+1,i,j})} \right),
\end{aligned}$$

where

$$\lambda'_{\ell-1,i}(x,y) = \lambda_{\ell-1,i}(x) - \phi_{\boldsymbol{\theta}_{(\ell,k)\to(\ell-1,i)}}(x-y)$$

, and

$$\tilde{\lambda}'_{\ell+1,i}(x,y) = \tilde{\lambda}_{\ell+1,i}(x) - \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,k)\to(\ell+1,i)}}(y-x).$$

So the acceptance probability is

$$\alpha = \min(1, \mathcal{P}_r).$$

C.4 Swap a Real Event and a Virtual Event

Suppose we want to flip a virtual event $\tilde{t}_{\ell,k,j}$ to a real event $t_{\ell,k,m_{\ell,i}+1}$ and we also want to flip a real event $t_{\ell,k,j}$ to a virtual event $\tilde{t}_{\ell,k,\tilde{m}_{\ell,k}+1}$.

The probability to pick $\tilde{t}_{\ell,k,j}$ and $t_{\ell,k,j}$ is $1/m_{\ell,k} \cdot 1/\tilde{m}_{\ell,k}$, the same for the probability to pick the real and virtual events in the reverse move. Thus, the proposal probability ratio is $\mathcal{Q} = 1$.

The likelihood ratio is

$$\mathcal{P} = \mathcal{P}_v \cdot \mathcal{P}_r\ .$$

So the acceptance probability is

$$\alpha = \min(1, \mathcal{P}).$$

Appendix D

Optimization

D.1 Derivatives *w.r.t* the Parameters for VPPs

For simplicity, we omit the data index n. The likelihood of the parameters for the VPPs $w.r.t$ the real events is

$$\tilde{\text{llh}} = \sum_{\ell=1}^{L}\sum_{k=1}^{K_\ell}\left(\sum_{j=1}^{\tilde{m}_{\ell,k}} \log \tilde{\lambda}_{\ell,k}(t_{\ell,k,j}) - \int_0^T \tilde{\lambda}_{\ell,k}(t)dt\right),$$

where

$$\int_0^T \tilde{\lambda}_{\ell,k}(t)dt = \tilde{\mu}_{\ell,k}T + \sum_{i=1}^{K_{\ell-1}}\sum_{j=1}^{m_{\ell-1,i}} \tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t_{\ell-1,i,j}).$$

The integral of the virtual kernel function is

$$\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{(\ell-1,i)\to(\ell,k)}}(t) = \int_0^t \tilde{\phi}_{\tilde{\boldsymbol{\theta}}_{(\ell,i)\to(\ell+1,k)}}(\tau)d\tau = \int_0^t \tilde{p}\frac{\tilde{\beta}^{\tilde{\alpha}}}{\Gamma(\tilde{\alpha})}\tau^{\tilde{\alpha}-1}e^{-\tilde{\beta}\tau}d\tau = \tilde{p}\frac{1}{\Gamma(\tilde{\alpha})}\gamma(\tilde{\alpha},\tilde{\beta}t)$$

with $\tilde{p} = \tilde{p}_{(\ell-1,i)\to(\ell,k)}$, $\tilde{\alpha} = \tilde{\alpha}_{(\ell-1,i)\to(\ell,k)}$, $\tilde{\beta} = \tilde{\beta}_{(\ell-1,i)\to(\ell,k)}$.

The partial derivative of $\tilde{\text{llh}}$ $w.r.t$ $\tilde{\mu}_{\ell,k}$ is

$$\partial_{\tilde{\mu}_{\ell,k}}\tilde{\text{llh}} = \sum_{j=1}^{\tilde{m}_{\ell,k}} \frac{1}{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})} - T.$$

In the following section, we use $\tilde{\boldsymbol{\theta}}$ to denote $\tilde{\boldsymbol{\theta}}_{(\ell-1,*)\to(\ell,k)}$.

The partial derivative of $\tilde{\mathrm{llh}}$ *w.r.t* $\tilde{p}$ is

$$\begin{aligned}\partial_{\tilde{p}}\tilde{\mathrm{llh}} =&\partial_{\tilde{p}}\left(\sum_{j=1}^{\tilde{m}_{\ell,k}}\log\tilde{\lambda}_{\ell,k}(t_{\ell,k,j}) - \sum_{i=1}^{K_{\ell-1}}\sum_{j=1}^{m_{\ell-1,i}}\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}}(t_{\ell-1,i,j})\right)\\ =&\sum_{j=1}^{\tilde{m}_{\ell,k}}\frac{\partial_{\tilde{p}}\phi_{\tilde{\boldsymbol{\theta}}_{\tilde{\boldsymbol{\theta}}}}(t_{\ell,k,j})}{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})} - \sum_{i=1}^{K_{\ell-1}}\sum_{j=1}^{m_{\ell-1,i}}\partial_{\tilde{p}}\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}_{\tilde{\boldsymbol{\theta}}}}(t_{\ell-1,i,j}),\end{aligned}$$

where

$$\partial_{\tilde{p}}\phi_{\tilde{\boldsymbol{\theta}}}(t) = \frac{\tilde{\beta}^{\tilde{\alpha}}}{\Gamma(\tilde{\alpha})}t^{\tilde{\alpha}-1}e^{-\tilde{\beta}t},\ \partial_{\tilde{p}}\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}}(t) = \frac{1}{\Gamma(\tilde{\alpha})}\gamma(\tilde{\alpha},\tilde{\beta}t).$$

The partial derivative of $\tilde{\mathrm{llh}}$ *w.r.t* $\tilde{\alpha}$ is

$$\partial_{\tilde{\alpha}}\tilde{\mathrm{llh}} = \sum_{j=1}^{\tilde{m}_{\ell,k}}\frac{\partial_{\tilde{\alpha}}\phi_{\tilde{\boldsymbol{\theta}}}(t_{\ell,k,j})}{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})} - \sum_{i=1}^{K_{\ell-1}}\sum_{j=1}^{m_{\ell-1,i}}\partial_{\tilde{\alpha}}\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}}(t_{\ell-1,i,j}),$$

where

$$\begin{aligned}\partial_{\tilde{\alpha}}\phi_{\tilde{\boldsymbol{\theta}}}(t) =&\tilde{p}\frac{(\tilde{\beta}t)^{\tilde{\alpha}-1}\ln(\tilde{\beta}t)\Gamma(\tilde{\alpha}) - (\tilde{\beta}t)^{\tilde{\alpha}-1}\Psi(\tilde{\alpha})\Gamma(\tilde{\alpha})}{\Gamma^2(\tilde{\alpha})}\tilde{\beta}e^{-\tilde{\beta}t} = \tilde{p}(\tilde{\beta}t)^{\tilde{\alpha}-1}\frac{\ln(\tilde{\beta}t)-\Psi(\tilde{\alpha})}{\Gamma(\tilde{\alpha})}\tilde{\beta}e^{-\tilde{\beta}t},\\ \partial_{\tilde{\alpha}}\Phi_{\tilde{\boldsymbol{\theta}}}(t) =&\tilde{p}\left(-\frac{\Psi(\tilde{\alpha})}{\Gamma(\tilde{\alpha})}\gamma(\tilde{\alpha},\tilde{\beta}t) + \frac{1}{\Gamma(\tilde{\alpha})}\frac{\partial\gamma(\tilde{\alpha},\tilde{\beta}t)}{\partial\tilde{\alpha}}\right)\\ =&\tilde{p}\left(-\frac{\Psi(\tilde{\alpha})}{\Gamma(\tilde{\alpha})}\gamma(\tilde{\alpha},\tilde{\beta}t) + \frac{1}{\Gamma(\tilde{\alpha})}\frac{\partial(\Gamma(\tilde{\alpha})-\Gamma(\tilde{\alpha},\tilde{\beta}t))}{\partial\tilde{\alpha}}\right)\\ =&\tilde{p}\Bigg(-\frac{\Psi(\tilde{\alpha})}{\Gamma(\tilde{\alpha})}\gamma(\tilde{\alpha},\tilde{\beta}t) + \frac{1}{\Gamma(\tilde{\alpha})}\cdot\Big(\Psi(\tilde{\alpha})\Gamma(\tilde{\alpha}) - \ln(\tilde{\beta}t)\Gamma(\tilde{\alpha},\tilde{\beta}t) - \tilde{\beta}t\cdot T(3,\tilde{\alpha},\tilde{\beta}t)\Big)\Bigg),\end{aligned}$$

where $T(m,s,x)$ a special case of the Meijer G-function

$$T(m,s,x) = G_{m-1,m}^{m,0}\left(\left.\begin{matrix}\underbrace{0,0,\cdots,0}_{m-1}\\ \\ \underbrace{s-1,-1,\cdots,-1}_{m}\end{matrix}\right| x\right).$$

However, it is expensive and numerically unstable to directly calculate Meijer G-function, so we use the first order finite difference to approximate the derivative.

The partial derivative of $\tilde{\text{llh}}$ *w.r.t* $\tilde{\beta}$ is

$$\partial_{\tilde{\beta}}\tilde{\text{llh}} = \sum_{j=1}^{\tilde{m}_{\ell,k}} \frac{\partial_{\tilde{\beta}}\phi_{\tilde{\boldsymbol{\theta}}}(t_{\ell,k,j})}{\tilde{\lambda}_{\ell,k}(t_{\ell,k,j})} - \sum_{i=1}^{K_{\ell-1}}\sum_{j=1}^{m_{\ell-1,i}} \partial_{\tilde{\beta}}\tilde{\Phi}_{\tilde{\boldsymbol{\theta}}}(t_{\ell-1,i,j}),$$

where

$$\partial_{\tilde{\beta}}\phi_{\tilde{\boldsymbol{\theta}}}(t) = \tilde{p}/\Gamma(\tilde{\alpha})t^{\tilde{\alpha}-1}(\tilde{\alpha}\tilde{\beta}^{\tilde{\alpha}-1}e^{-\tilde{\beta}t} - \tilde{\beta}^{\tilde{\alpha}}e^{-\tilde{\beta}t}t) = \tilde{p}/\Gamma(\tilde{\alpha})t^{\tilde{\alpha}-1}\tilde{\beta}^{\tilde{\alpha}-1}e^{-\tilde{\beta}t}(\tilde{\alpha}-\tilde{\beta}t)\ ,$$

$$\partial_{\tilde{\beta}}\Phi_{\tilde{\boldsymbol{\theta}}}(t) = \tilde{p}\frac{1}{\Gamma(\tilde{\alpha})}\frac{\partial\gamma(\tilde{\alpha},\tilde{\beta}t)}{\partial\tilde{\beta}} = \tilde{p}\frac{1}{\Gamma(\tilde{\alpha})}(\tilde{\beta}t)^{\tilde{\alpha}-1}e^{-\tilde{\beta}t}\cdot t\ .$$

We use softplus function to make sure the parameters are all positive.

D.2 Maximization and Derivatives *w.r.t* the Parameters for RPPs

The likelihood of the parameters for $\mathbf{Z}_n$ *w.r.t* the real events is

$$\text{llh} = \sum_{\ell=0}^{L}\sum_{k=1}^{K_\ell}\left(\sum_{j=1}^{m_{n,\ell,k}} \log\lambda_{n,\ell,k}(t_{n,\ell,k,j}) - \int_0^T \lambda_{n,\ell,k}(t)dt\right),$$

where

$$\int_0^T \lambda_{n,L,k}(t)dt = \mu_{n,k}T\ ,$$

$$\int_0^T \lambda_{n,\ell,k}(t)dt = \sum_{i=1}^{K_{\ell+1}}\sum_{j=1}^{m_{n,\ell+1,i}} \Phi_{\boldsymbol{\theta}_{(\ell+1,i)\to(\ell,k)}}(t_{n,\ell+1,i,j}) \text{ for } 0 \le \ell \le L-1\ .$$

The maximizing value for $\mu_{n,L,k}$ is

$$\mu_{n,L,k} = \frac{m_{n,L,k}}{T}.$$

The functional forms for the derivatives of the other parameters of the kernel functions are the same as the forms for VPPs.

Appendix E

Evidence Lower Bound

E.1 Proof of Proposition 5.2

Proposition 5.2. *Minimizing the exclusive KL divergence* $D_{KL}(Q(\mathbf{Z}^E) \parallel P(\mathbf{Z}))$ *is equivalent to maximizing the evidence lower bound (ELBO)*

$$\begin{aligned}\mathcal{L} = &- \sum_{\ell,i} \int_{S_{\ell,i}} \lambda^E_{\ell,i}(t)(\log \lambda^E_{\ell,i}(t) - 1)dt \\ &+ \Bigg(\sum_{t^x_{0,k,j}} \int \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h}) dQ(\mathbf{Z}^E) \\ &\quad - \sum_{\ell\geq 1,i,k} \int_{S_{\ell,i}} \Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}}(t)\lambda^E_{\ell,i}(t)dt \\ &\quad + \sum_{\ell\geq 1} \int \sum_{k,j} \log \sum_{i,h} \phi_{(\ell+1,i)\to(\ell,k)}(t^z_{\ell,k,j} - t^z_{\ell+1,i,h}) dQ(\mathbf{Z}^E) \Bigg),\end{aligned} \tag{5.4}$$

where $\{S_{\ell,i}\}$ *are the bounded regions where the point processes exist,* $\Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}} = \int_{S_{\ell-1,k}} \phi_{(\ell,i)\to(\ell-1,k)}(\tau - t)d\tau$*,* t^x *represents an event from the data, and* t^z *represents an event from the approximate point process. Moreover,* $\mathcal{L}$ *is the lower bound for the marginal probability density function.*

Proof. We choose the Poisson process with 1 as its intensity function as our reference point process. Denote the probability measure for the reference point process as P_0.

Then according to Definition 5.1 and Proposition 1.4,

$$
\begin{aligned}
& D_{KL}(Q(\mathbf{Z}^E) \parallel P(\mathbf{Z} \mid \mathbf{x})) && \text{(E.1)}\\
=& \int \log \left(\frac{dQ(\mathbf{Z}^E)}{dP(\mathbf{Z} \mid \mathbf{x})} \right) dQ(\mathbf{Z}^E) && \text{(E.2)}\\
=& \int \log \left(\frac{dQ(\mathbf{Z}^E)}{dP_0} \cdot \frac{dP_0}{dP(\mathbf{Z} \mid \mathbf{x})} \right) dQ(\mathbf{Z}^E) && \text{(E.3)}\\
=& \int \log \left(\frac{dQ(\mathbf{Z}^E)}{dP_0} \right) dQ(\mathbf{Z}^E) - \int \log \left(\frac{dP(\mathbf{Z} \mid \mathbf{x})}{dP_0} \right) dQ(\mathbf{Z}^E). && \text{(E.4)}
\end{aligned}
$$

The first part of Equation E.4 is

$$
\begin{aligned}
& \int \log \left(\frac{dQ(\mathbf{Z}^E)}{dP_0} \right) dQ(\mathbf{Z}^E) && \text{(E.5)}\\
=& \int \left(- \sum_{\ell,i} \int_{S_{\ell,i}} (\lambda^E_{\ell,i}(t) - 1) dt + \sum_{\ell,i,j} \log \lambda^E_{\ell,i}(t^z_{\ell,i,j}) \right) \; dQ(\mathbf{Z}^E) && \text{(E.6)}\\
=& \sum_{\ell,i} \int_{S_{\ell,i}} \lambda^E_{\ell,i}(t)(\log \lambda^E_{\ell,i}(t) - 1) dt + \sum_{\ell,i} \int_{S_{\ell,i}} dt. && \text{(E.7)}
\end{aligned}
$$

The second part of Equation E.4 is

$$
\begin{aligned}
& \int \log \left(\frac{dP(\mathbf{Z} \mid \mathbf{x})}{dP_0} \right) dQ(\mathbf{Z}^E) && \text{(E.8)}\\
=& \int \log f(\mathbf{Z} \mid \mathbf{x}) dQ(\mathbf{Z}^E) && \text{(E.9)}\\
=& \int \log f(\mathbf{Z}, \mathbf{x}) dQ(\mathbf{Z}^E) - \log f(\mathbf{x}), && \text{(E.10)}
\end{aligned}
$$

where

$$
\begin{aligned}
& \int \log f(\mathbf{Z}, \mathbf{x}) dQ(\mathbf{Z}^E) = \int \log(f(\mathbf{x} \mid \mathbf{Z}) \cdot f(\mathbf{Z})) dQ(\mathbf{Z}^E) && \text{(E.11)}\\
=& \int \Bigg(- \sum_{i,j,k} \int_{S_{0,k}} (\phi_{(1,i)\to(0,k)}(t - t^z_{1,i,j}) - 1) dt + \sum_{t^x_{0,k,j}} \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h})
\end{aligned}
$$

$$- \sum_{\ell \geq 1, k} \int_{S_{\ell,k}} (\lambda^p_{\ell,k}(t) - 1)dt + \sum_{\ell \geq 1,k,j} \log \lambda^p_{\ell,k}(t^z_{\ell,k,j}) \Bigg) dQ(\mathbf{Z}^E) \tag{E.12}$$

$$= \int \Bigg(- \sum_{i,j,k} \int_{S_{0,k}} (\phi_{(1,i)\to(0,k)}(t - t^z_{1,i,j}) - 1)dt + \sum_{t^x_{0,k,j}} \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h})$$

$$- \sum_{\ell \geq 1,k} \int_{S_{\ell,k}} (\sum_{i,j} \phi_{(\ell+1,i)\to(\ell,k)}(t - t^z_{\ell+1,i,j}) - 1)dt$$

$$+ \sum_{\ell \geq 1,k,j} \log \sum_{i,h} \phi_{(\ell+1,i)\to(\ell,k)}(t^z_{\ell,k,j} - t^z_{\ell+1,i,h}) \Bigg) dQ(\mathbf{Z}^E) \tag{E.13}$$

$$= \sum_{t^x_{0,k,j}} \int \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h}) dQ(\mathbf{Z}^E) - \sum_{i,k} \int_{S_{1,i}} \Phi_{(1,i)\to(0,k);S_{0,k}}(t) \lambda^E_{1,i}(t) dt$$

$$- \sum_{\ell \geq 1,i,k} \int_{S_{\ell+1,i}} \Phi_{(\ell+1,i)\to(\ell,k);S_{\ell,k}}(t) \lambda^E_{\ell+1,i}(t) dt$$

$$+ \sum_{\ell \geq 1} \int \sum_{k,j} \log \sum_{i,h} \phi_{(\ell+1,i)\to(\ell,k)}(t^z_{\ell,k,j} - t^z_{\ell+1,i,h}) dQ(\mathbf{Z}^E) + \sum_{\ell,k} \int_{S_{\ell,k}} dt \tag{E.14}$$

$$= \sum_{t^x_{0,k,j}} \int \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h}) dQ(\mathbf{Z}^E) \tag{E.15}$$

$$- \sum_{\ell \geq 1,i,k} \int_{S_{\ell,i}} \Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}}(t) \lambda^E_{\ell,i}(t) dt$$

$$+ \sum_{\ell \geq 1} \int \sum_{k,j} \log \sum_{i,h} \phi_{(\ell+1,i)\to(\ell,k)}(t^z_{\ell,k,j} - t^z_{\ell+1,i,h}) dQ(\mathbf{Z}^E) + \sum_{\ell,k} \int_{S_{\ell,k}} dt. \tag{E.16}$$

Combining Equations E.7, E.10 and E.4, the KL divergence becomes

$$D_{KL}(Q(\mathbf{Z}^E) \parallel P(\mathbf{Z} \mid \mathbf{x})) \tag{E.17}$$

$$= \sum_{\ell,i} \int_{S_{\ell,i}} \lambda^E_{\ell,i}(t)(\log \lambda^E_{\ell,i}(t) - 1)dt + \sum_{\ell,i} \int_{S_{\ell,i}} dt$$

$$- \Bigg(\sum_{t^x_{0,k,j}} \int \log \sum_{t^z_{1,i,h}} \phi_{(1,i)\to(0,k)}(t^x_{0,k,j} - t^z_{1,i,h}) dQ(\mathbf{Z}^E)$$

$$- \sum_{\ell \geq 1,i,k} \int_{S_{\ell,i}} \Phi_{(\ell,i)\to(\ell-1,k);S_{\ell-1,k}}(t) \lambda^E_{\ell,i}(t) dt$$

$$+\sum_{\ell\geq 1}\int\sum_{k,j}\log\sum_{i,h}\phi_{(\ell+1,i)\to(\ell,k)}(t^{z}_{\ell,k,j}-t^{z}_{\ell+1,i,h})dQ(\mathbf{Z}^{E})+\sum_{\ell,k}\int_{S_{\ell,k}}dt\Bigg)+\log f(\mathbf{x}) \tag{E.18}$$

$$=-\mathcal{L}+\log f(\mathbf{x}) \tag{E.19}$$

Thus, if the parameters for the kernels are fixed, then minimizing the KL divergence is equivalent to maximizing the ELBO $\mathcal{L}$.

Moreover, since $\mathcal{L}=\log f(\mathbf{x})-D_{KL}(Q(\mathbf{Z}^{E})\parallel P(\mathbf{Z}\mid\mathbf{x}))$ and the KL divergence is nonnegative, $\mathcal{L}$ is the lower bound of the marginal probability density function. Then, we can maximize the marginal probability indirectly by maximizing the ELBO $\mathcal{L}$. □

Appendix F

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 $\boldsymbol{pe}_{\ell,i,j}$ though positional encoding,

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

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

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 $x^e_{\ell,i,j}$ of each event $e_{\ell,i,j}$ can be represented as

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

Self-Attention The parameters of the intensity function $\lambda^I_{\ell,k}(t)$ 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,

$$\begin{aligned} \boldsymbol{h}^a_{\ell-1,i,j} = & \left(\sum_{t=1}^{m_{\ell-1,i}} f\left(W_q \cdot \text{LayerNorm}(\boldsymbol{x}^e_{\ell-1,i,j}), W_k \cdot \boldsymbol{x}^e_{\ell-1,i,t}\right) \cdot \left(W_v \cdot \boldsymbol{x}^e_{\ell-1,i,t}\right) \right) \\ & / \sum_{t=1}^{m_{\ell-1,i}} f\left(W_q \cdot \text{LayerNorm}(\boldsymbol{x}^e_{\ell-1,i,j}), W_k \cdot \boldsymbol{x}^e_{\ell-1,i,t}\right), \end{aligned} \tag{F.1}$$

where 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 $f(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(\boldsymbol{x}_1^T \boldsymbol{x}_2)$ 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 (Proposition 4.1).

Equation F.1 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 Equation F.1 as

$$\boldsymbol{h}^a_{\ell-1,i,j} = \text{Self-Attention}(\boldsymbol{x}^e_{\ell-1,i,j}, W_q, W_k, W_v),$$

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

$$\begin{aligned}\boldsymbol{h}^a_{\ell-1,i,j} =& \text{Multi-Head-Self-Attention}(\boldsymbol{x}^e_{\ell-1,i,j}) \\ =& \text{Concat}(\text{head}_1, \cdots, \text{head}_h) \cdot W_o,\end{aligned} \tag{F.2}$$

where $\text{head}_d = \text{Self-Attention}(\boldsymbol{x}^e_{\ell-1,i,j}, W^d_q, W^d_k, W^d_v)$, $W^d_q \in \mathbb{R}^{d_k \times d_M}$, $W^d_k \in \mathbb{R}^{d_k \times d_M}$, $W^d_v \in \mathbb{R}^{d_v \times d_M}$, 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 $\boldsymbol{h}^a_{\ell-1,i,j}$ and the residual $\boldsymbol{x}^e_{\ell-1,i,j}$ 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,

$$\boldsymbol{h}_{\ell-1,i,j} = \left(W^F_2(\text{GELU}(W^F_1(\text{LayerNorm}(\text{input})) + \boldsymbol{b}^F_1)) + \boldsymbol{b}^F_2\right) + \text{input}, \tag{F.3}$$

where $\text{input} = \boldsymbol{h}^a_{\ell-1,i,j} + \boldsymbol{x}^e_{\ell-1,i,j}$, $W^F_1 \in \mathbb{R}^{d_H \times d_M}$, $\boldsymbol{b}^F_1 \in \mathbb{R}^{d_H \times 1}$, $W^F_2 \in \mathbb{R}^{d_M \times d_H}$, $\boldsymbol{b}^F_2 \in \mathbb{R}^{d_M \times 1}$, 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,

$$\boldsymbol{\theta}^I_{\ell,k} = W^\theta_{\ell,k} h_{\ell-1,i,j} + \boldsymbol{b}^\theta_{\ell,k},$$

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