Generative Sequential Stochastic Model for Marked Point Processes

Abhishek Sharma [*1] **Aritra Ghosh** [*1] **Madalina Fiterau** [1]

Abstract

Temporal point processes are often used to model event data streams in the real world, such as financial transactions, electronic health records, and social networks. While several parametric approaches, such as Poisson and Hawkes processes are used to model temporal dynamics, they often suffer from the curse of model misspecification. More flexible RNN based approaches have been recently proposed to jointly model event time and marker information through the use of a shared hidden representation. Although the RNN based approaches achieve superior results compared to the earlier approaches, all these methods make the restrictive assumption that the event and marker dynamics follow the same distribution for all subpopulations. To do away with this assumption, we propose a stochastic model combining subpopulation cluster dynamics with the recurrent marked point process. We posit that this generative model can better capture the dynamics of event states conditioned on the discrete and continuous latent structure. We validate this hypothesis by evaluating this model on synthetic as well as several real-world datasets.

1. Introduction

Temporal dataset of event streams can be represented as $\mathcal{S} = \{(\mathbf{t}^j_{1:T_j}, \mathbf{x}^j_{1:T_j})_{j=1}^N\}$, where $\mathbf{x}^j_{1:T} = (\mathbf{x}^j_1, \cdots, \mathbf{x}^j_{T_j})$ are the marker or event information and $\mathbf{t}^j_{1:T_j} = (\mathbf{t}^j_1, \cdots, \mathbf{t}^j_{T_j}), \mathbf{t}^j_i < \mathbf{t}^j_{i+1}$, are the real valued timestamps for the event sequences, for sequence j of length T_j. The event markers can either be a real valued (bio-markers for patients visiting hospital) or a categorical (buy/sell events in financial transactions) random variable. In this work, we address the problem of modeling such datasets using *temporal point processes*.

[*]Equal contribution [1]College of Information and Computer Sciences, University of Massachusetts, Amherst. Correspondence to: Aritra Ghosh <arighosh@cs.umass.edu>.

Presented at the ICML 2019 Time Series Workshop. Copyright 2019 by the author(s).

Temporal point processes are often used to model the density of time of the next event conditioned on the history, i.e. $f(\mathbf{t}_n \mid \mathcal{H}_{\mathbf{t}_n})$, $\mathcal{H}_{\mathbf{t}_n} := \{\mathbf{t}_{1:n-1}\}$ (Rasmussen, 2011). Typically, point processes parameters are encoded using the *conditional intensity function*, $\lambda^*(\mathbf{t}) = \lambda(\mathbf{t}|\mathcal{H}_{<\mathbf{t}}) = \Pr[\text{event} \in [\mathbf{t}, \mathbf{t} + d\mathbf{t})|\mathcal{H}_{<\mathbf{t}}]$, which given the past, encodes the probability of an event occuring at future time points. For example, conditional Intensity function of the Hawkes process is defined as, $\lambda(\mathbf{t}) = \lambda_0 + \alpha \sum_{\mathbf{t}_i < t} \exp(-\frac{\mathbf{t}-\mathbf{t}_i}{\sigma})$, for $\lambda_0, \alpha, \sigma > 0$. This equation asserts that the probability of an event occurring increases right after the previous event (self-exciting), and decays exponentially over time otherwise. When a property of the dynamics is known (e.g., self-exciting events, self-correcting events, etc.), several classic temporal point process models, such as Poisson and Hawkes processes, work well (Hawkes & Oakes, 1974). However, they are often too simplistic when applied to large real-world datasets. Du et al. (2016) propose an alternative to this parameterization, summarizing the history $\mathcal{H}_{t_n}$ using the hidden state of a recurrent neural network. Du et al. (2016) proposed to model the intensity function of the point process using the following exponentiated linear function, where $\mathbf{h}$ represents the hidden state of RNN with the sequence as inputs,

$$\lambda^*(\mathbf{t}) = \exp(v^{\mathbf{t}^T}\mathbf{h}_j + \mathbf{w}^T(\mathbf{t} - \mathbf{t}_j) + b_t) \tag{1}$$

The time likelihood of n^{th} event at time $\mathbf{t}$, given the history is simply $f^*(t) = \lambda^*(t) \exp\left(-\int_{t_{n-1}}^t \lambda^*(\tau) d\tau\right)$. The linear form in Eq. 1 allows calculating the time log-likelihood in analytical form. We can also choose to directly model time probability density function using an arbitrary non-linear parameterization (Xiao et al., 2017c). In Section 2, we discuss our model agnostic to any choice of time/intensity density parameterization.

Following the success of the recurrent approach to model point processes, several neural network-based algorithms have been proposed. In an attempt to generalize these processes, Mei & Eisner (2017) proposed the Neural Hawkes process using a modified *continuous time LSTM cell*. To further incorporate an inherent clustering in the sequence of dynamics, a mixture of Hawkes processes was proposed (Xu & Zha, 2017; Du et al., 2015). However, real-world data is often generated by a mixture of diverse dynamic processes, and an algorithm that learns only certain types of tempo-

ral processes can lead to model bias. For example, in the clinical setting, the health status of a patient with diabetes may decline over time (self-exciting events), while a patient with hypertension may recover over time (self-correcting events) implicating separate point process dynamics within the same electronic health records dataset.

Furthermore, real-world stochasticity is often ignored while modeling point processes. This has been the focus of recent work on augmenting a deep state-space model with a recurrent neural network to effectively model sequential data (Fraccaro et al., 2016; Krishnan et al., 2017). We propose a stochastic sequential generative model for point processes augmented by recurrent deterministic hidden states, stochastic sequential states, and categorical latent states to capture mixtures of different point processes while learning useful cluster representations among the sample dynamics.

In this work, we make the following contributions: we propose two generative models that try to capture the dynamics of marked point processes. These models aim to capture mixtures of point processes that could be generating the data using additional stochastic latent states.

2. Methods

We define a deep generative model, p_θ, with deterministic hidden recurrent states, $\mathbf{h}_{1:T}$, stochastic continuous states, $\mathbf{z}_{1:T}$, as well as a categorical discrete latent state, $\mathbf{y}$, on the temporal dynamics. In the subsequent discussion, we will use $\vec{\mathbf{a}}$ as a shorthand for $\mathbf{a}_{1:T}$ and use $A = \{\mathbf{z}, \mathbf{h}\}_0$ to denote initial states. With this overall structure, we propose two models to capture the latent dynamics of the marked temporal point process. In Figure 1, the generative and inference modules are shown for both the models.

2.1. Generative Model

We can write the joint distribution of the data, the continuous states ($\mathbf{z}_{1:T}$), the deterministic states ($\mathbf{h}_{1:T}$), and the discrete categorical prior ($\mathbf{y}$) in the graphical Model 1 (Figure 1a) as:

$$p_\theta(\vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y}|A) = p_{\theta_\mathbf{y}}(\mathbf{y}) \prod_{i=1}^{T} p_{\theta_\mathbf{z}}(\mathbf{z}_i) p_{\theta_{\mathbf{x},\mathbf{t}}}(\mathbf{x}_i, \mathbf{t}_i|\mathbf{y}, \mathbf{z}_i, \mathbf{h}_i) \times p_{\theta_\mathbf{h}}(\mathbf{h}_i|\mathbf{x}_{i-1}, \mathbf{t}_{i-1}, \mathbf{h}_{i-1})$$

while for Model 2 (Figure 1c), the joint $p_\theta(\vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y}|A)$ factorizes as:

$$p_{\theta_\mathbf{y}}(\mathbf{y}) \prod_{i=1}^{T} p_{\theta_\mathbf{z}}(\mathbf{z}_i|\mathbf{z}_{i-1}) p_{\theta_\mathbf{h}}(\mathbf{h}_i|(\mathbf{x}, \mathbf{t}, \mathbf{h})_{i-1}) \times p_{\theta_{\mathbf{x},\mathbf{t}}}((\mathbf{x}, \mathbf{t})_i|\mathbf{y}, (\mathbf{z}, \mathbf{h})_i)$$

where, the latent categorical variable, $\mathbf{y}$, taking K discrete states, has density $p_{\theta_\mathbf{y}}(\mathbf{y}) = \prod_{k=1}^{K} p_{\theta_\mathbf{y}}(\mathbf{y}_k = 1)$, and parameters $\theta_\mathbf{y}$. The deterministic states are parameterized using an RNN (or modern variants like GRU, LSTM) with parameters $\theta_\mathbf{h}$. Furthermore, to allow for stochasticity in the data generation dynamics, the variables $\vec{\mathbf{z}}$ influence the observed variables through a *stochastic layer* with parameters $\theta_\mathbf{z}$. The states $\{\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y}\}$ jointly generate the observed data, which are event markers ($\vec{\mathbf{x}}$) and event timestamps ($\vec{\mathbf{t}}$). We define $\theta = \{\theta_{\mathbf{x},\mathbf{t}}, \theta_\mathbf{z}, \theta_\mathbf{y}, \theta_\mathbf{h}\}$ as the generative model parameters. There are multiple choices for parameterization of $\theta_{\mathbf{x},\mathbf{t}}$. To model event time information, we can have both a Normal distribution as well as a parametric conditional intensity function (equation 1). To predict the categorical event markers, we learn the logits of the distribution.

The two models induce two variants in the dynamics of stochastic sequential states. Consequently, we have separate assumptions on the priors of latent variables. For the first variant, each stochastic state prior is assumed to be Normally distributed, $p_{\theta_\mathbf{z}}(\mathbf{z}_i) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Thus an independent stochastic perturbation is applied to the point process at each time step. In the second variant, we model $p_{\theta_\mathbf{z}}(\mathbf{z}_{0:T})$ using a Deep Markov Model (Krishnan et al., 2017), leading us to assume $p_{\theta_\mathbf{z}}(\mathbf{z}_i|\mathbf{z}_{0:i-1}) = \mathcal{N}(\mathbf{z}_i; \mu_i(\mathbf{z}_{i-1}), \sigma_i^2(\mathbf{z}_{i-1}))$, where the transition matrix, is implemented using a non-linear neural networks. The initial stochastic state is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. We use a uniform categorical prior on the discrete latent states, $\mathbf{y}$.

The log-likelihood of observed event and time sequence of the samples $\mathcal{S}$ is,

$$\mathcal{L}(\theta) = \sum_i \log p_\theta(\mathbf{x}^i_{1:T_i}, \mathbf{t}^i_{1:T_i}|\mathbf{z}_0, \mathbf{h}_0) = \sum_i \mathcal{L}_i(\theta)$$

To aid readability, we omit the sample index i from the equations to whenever it is clear from context. To compute the log-likelihood, we need to marginalize the variables $\vec{\mathbf{z}}$, $\mathbf{y}$ and $\vec{\mathbf{h}}$. The distribution of deterministic states $\mathbf{h}_t$ can be thought as a dirac delta function centered at $\tilde{\mathbf{h}}_t = f_{\theta_h}(\tilde{\mathbf{h}}_{t-1}, \mathbf{x}_{t-1}, \mathbf{t}_{t-1})$ and $\theta_{\tilde{\mathbf{h}}}$ being the RNN parameters. Thus by replacing $\mathbf{h}_{1:T}$ with $\tilde{\mathbf{h}}_{1:T}$, we can marginalize $\mathbf{h}$ (Fraccaro et al., 2016). The marginalization of the discrete state, as well as sequential continuous states, can be approximated using ancestral Monte Carlo sampling. We discuss this in the subsequent section.

2.2. Inference Model

To maximize log-likelihood of the observed data, stochastic sequential states as well as the discrete latent states need to be marginalized out. Due to non-linear nature of the parameterization, analytically marginalizing out the variables is not possible. Furthermore, for the non-linearly dependent categorical discrete latent variable, exact inference would increase computation as $\mathcal{O}(K)$ where K is the number of states the variable can take. For large enough datasets, even

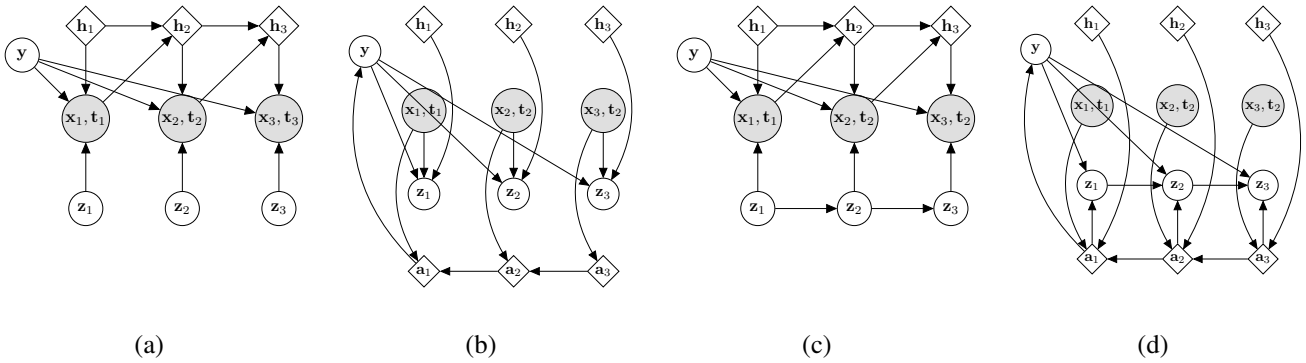

Figure 1. (a) Generative Model for Model 1 (b) Inference Model for Model 1, (c) Generative Model for Model 2 (d) Inference Model for Model 2. $\mathbf{z}_i$ are stochastic sequential nodes. $\mathbf{y}$ is categorical latent state. $\mathbf{h}$ and $\mathbf{a}$ are deterministic hidden states of forward and reverse RNN. $\mathbf{x}, \mathbf{t}$ are observation of the marked temporal point process.

the linear increase in computation is impractical. Thus, we resort to variational inference, where we maximize the variational evidence lower bound (ELBO), $\mathcal{F}(\theta, \phi)$, to the likelihood. We estimate the posterior distribution $p_\theta(\vec{\mathbf{z}}, \mathbf{y}, \vec{\mathbf{h}} | \vec{\mathbf{x}}, \vec{\mathbf{t}})$ with an *approximate posterior* $q_\phi(\vec{\mathbf{z}}, \mathbf{y}, \vec{\mathbf{h}} | \vec{\mathbf{x}}, \vec{\mathbf{t}})$. Thus the log-likelihood can be written as [1]:

$$\begin{aligned}\mathcal{L}(\theta) \geq & \mathcal{F}(\theta, \phi) \\ &= \int q_\phi(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A) \log \frac{p_\theta(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \vec{\mathbf{x}}, \vec{\mathbf{t}}, \mathbf{y} | A)}{q_\phi(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A)} d\mathbf{y} d\vec{\mathbf{z}} d\vec{\mathbf{h}}\end{aligned}$$

where ϕ are the parameters of the *variational bayes* inference network while θ are the parameters of the generative neural network. We can maximize $\{\theta, \phi\}$ using stochastic gradient ascent.

To efficiently perform inference, we factorize the joint in a way that exploits the temporal structure of the model. Also, we do the same for the posterior to get the following decomposition for Model 1:

$$p_\theta(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A) = p_\theta(\mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\tilde{\mathbf{h}}}) \prod_i p_\theta(\mathbf{z}_i | (\mathbf{x}, \mathbf{t})_i, \mathbf{y}, \tilde{\mathbf{h}}_i)$$

We design our approximate posterior to take the same factorized form:

$$q_\phi(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A) = q_\phi(\mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\tilde{\mathbf{h}}}) \prod_i q_\phi(\mathbf{z}_i | (\mathbf{x}, \mathbf{t})_i, \mathbf{y}, \tilde{\mathbf{h}}_i)$$

where the deterministic hidden state $\tilde{\mathbf{h}}_i = f_{\theta_\mathbf{h}}(\tilde{\mathbf{h}}_{i-1}, \mathbf{x}_{i-1}, \mathbf{t}_{i-1})$ accounts for the effect of the past.

Similarly, for Model 2, the factorization of the posterior and its approximation are:

$$\begin{aligned}&p_\theta(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A) \\ &\quad = p_\theta(\mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\tilde{\mathbf{h}}}) \prod_{i=1}^{T} p_\theta(\mathbf{z}_i | \mathbf{z}_{i-1}, (\mathbf{x}, \mathbf{t})_{i:T}, \mathbf{y}, \tilde{\mathbf{h}}_{i:T}) \\ &q_\phi(\vec{\mathbf{z}}, \vec{\mathbf{h}}, \mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, A) \\ &\quad = q_\phi(\mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\tilde{\mathbf{h}}}) \prod_{i=1}^{T} q_\phi(\mathbf{z}_i | \mathbf{z}_{i-1}, (\mathbf{x}, \mathbf{t})_{i:T}, \mathbf{y}, \tilde{\mathbf{h}}_{i:T})\end{aligned}$$

Now we discuss how we sample from each of these distributions in the approximate posterior. Approximate posterior of the categorical latent variable, $q_\phi(\mathbf{y} | \vec{\mathbf{x}}, \vec{\mathbf{t}}, \vec{\tilde{\mathbf{h}}})$ is a Categorical distribution, parameterized by a neural network that takes in the observed variables and the hidden states as input. We accomplish this using a reverse RNN but choose to drop the dependence on $\vec{\tilde{\mathbf{h}}}$ for Model 1 to keep its inference much faster. The hidden state of the reverse RNN evolves as $\mathbf{a}_t = f_{\phi_\mathbf{h}}(\mathbf{a}_{t+1}, [\mathbf{x}_t, \mathbf{t}_t, \tilde{\mathbf{h}}_t])$. This is usually a fast and independent step, which only requires the knowledge of the inputs $\vec{\mathbf{x}_t}, \vec{\mathbf{t}}$ to compute. The inference network for continuous latent variables $\mathbf{z}_t$ is *amortized* over time and is modeled using a diagonal Gaussian: $z_i \sim \mathcal{N}(\mu_i^q, \log v_i^q)$. The parameters of this distribution have no temporal dependence in the case of Model 1, leading to much faster inference. In the case of Model 2, since there is temporal dependence of $\mathbf{z}_i$ on $\mathbf{z}_{i-1}$ (all the other inputs to this network can be computed beforehand), we need to perform inference in a serial manner.

We can sample a sequence $\vec{\mathbf{z}}$ using ancestral sampling, starting with sampling from $\mathbf{z}_1$, using the sampled $\mathbf{z}_{i-1}$ to obtain the next distribution. Each of these sampling steps is performed in a differentiable manner using the *reparameterization tick* (Kingma & Welling, 2013). Next, we discuss the

[1] Full derivation of ELBO for both models can be found in the Appendix

sampling of the categorical latent state, $\mathbf{y}$, which is also required for getting each of the $\mathbf{z}_i$. We found that the gradient estimate provided by the REINFORCE method (Williams, 2005) had a high variance for our experiments. Therefore we choose the Gumbel-softmax estimator (Jang et al., 2016), which provides a biased, but low variance estimator for the gradient, in practice.

Given the generative and inference network parameters $\{\theta, \phi\}$, we take a Monte Carlo sample using the reparameterization trick and maximize the ELBO [2] using stochastic gradient ascent.

3. Experiments

We evaluate our models on both synthetically generated and real-world datasets. For both cases, we report the log-likelihood (ELBO for our model which is a lower bound on the log-likelihood). The datasets we choose/generate differ greatly from each other in sequence-lengths, time-scales, and domains. We compare our results with the Recurrent Marked Temporal Point Process (RMTPP) model (Du et al., 2016). For all the experiments, we model the timestamp likelihood, $f^*(\mathbf{t})$, using a Gaussian distribution (Xiao et al., 2017c). Our inference network uses *smoothing*, thus the predictive metrics (e.g, the accuracy of next event, rmse of next event time) of future time steps are not valid. All models were implemented using PyTorch. All expectations were approximated (wherever necessary) using single samples to reduce runtime of the models.

3.1. Synthetic Data

To illustrate the effect of a mixture of multiple disparate point processes, we generate sequences from the Hawkes process, Homogenous Poisson Process, Non-homogenous Poisson process, and Self-correcting process. To generate a sequence from a process, we randomly sample k separate sets of pertinent parameters, where $k \in \{4, 8, 16\}$. For each set of k parameters, we sample 200 training sequences, 40 validation sequences and 50 test sequences. The mean and median sequence lengths for the training dataset is $\sim$ 35 and $\sim$ 20 respectively. We report the log-likelihood numbers in Table 1. Additional details about the synthetic data, parameters, hyper-parameter and network structure are provided in Appendix C.

3.2. Real World Data

We evaluate our model on five real-world datasets and report the results in Table 2. The MIMIC-II, Stack Overflow, and Financial datasets were processed as described in (Du et al., 2016), while the preprocessing of MemeTrack and Retweets datasets was identical to that of (Mei & Eisner, 2017).

[2] derived in the supplementary section

k	RMTPP	Model 1	Model 2
4	-0.0694	≥ 0.1311	$\geq$ **0.2285**
8	-0.0672	≥ 0.0003	$\geq$ **0.1863**
16	-0.1776	≥ 0.0209	$\geq$ **0.0445**

Table 1. Average log-likelihood for synthetic data for different sizes of parameter sets

From Table 2, we can observe that extra stochasticity (through latent states) can be powerful in modeling sequential behavior of marked point processes. MemeTracker and LastFM datasets contain 5000 and 3150 dimensional categorical markers, which makes it a difficult task to extract information from them to help the modeling process. We posit that this is a potential reason that the improvement in our model in these two datasets is small. MIMIC-II dataset also contains about 585 samples with 75 marker labels, making it difficult to effectively learn the behavior.

DATASET	RMTPP	Model 1	Model 2
MIMIC-II	-2.0668	≥ -1.8330	$\geq$ **-1.7462**
Stack Overflow	-5.1148	≥ -4.0039	$\geq$ **-3.3568**
MemeTrack	-2.5614	≥ -2.5930	$\geq$ **-2.5117**
Retweet	-1.2560	≥ -0.0765	$\geq$ **0.2707**
Financial	-0.4008	≥ 0.6687	$\geq$ **1.1937**
LastFM	-8.0050	≥ -7.9869	$\geq$ **-6.7539**

Table 2. Average log-likelihood for real-world datasets using Gaussian parameterization of time density

4. Discussion & Future Work

In this work, we propose a stochastic sequential model which combines both deep state space model as well as the deterministic RNN for modeling marked temporal point processes. To perform inference, we used *smoothing* techniques. However, in practical applications we are often interested in the *prediction* task, which leads us take up *filtering* as part of future work. With respect to the modeling decisions, instead of assuming conditionally independence for the marker and timestamp at a certain point in time, jointly modeling them dependent should make the model more effective. Finally, while using Gaussian distribution to model the time density allows us to parameterize any non-linear function, the conditional intensity function is more meaningful in case of point processes. Furthermore, modeling conditional intensity as a non-linear function also remains a challenge. We intend to compare these design choices as part of future work.

References

Bacry, E., Jaisson, T., and Muzy, J.-F. Estimation of slowly decreasing hawkes kernels: application to high-frequency order book dynamics. *Quantitative Finance*, 16(8):1179–1201, 2016.

Bayer, J. and Osendorfer, C. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Du, N., Song, L., Woo, H., and Zha, H. Uncover topic-sensitive information diffusion networks. In *Artificial Intelligence and Statistics*, pp. 229–237, 2013.

Du, N., Farajtabar, M., Ahmed, A., Smola, A. J., and Song, L. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 219–228. ACM, 2015.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. 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*, pp. 1555–1564. ACM, 2016.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential Neural Models with Stochastic Layers. 2016. ISSN 10495258. 10.1163/156853991X00490.

Goyal, A. G. A. P., Sordoni, A., Côté, M.-A., Ke, N. R., and Bengio, Y. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pp. 6713–6723, 2017.

Hawkes, A. G. and Oakes, D. A cluster process representation of a self-exciting process. *Journal of Applied Probability*, 11(3):493–503, 1974.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Li, Y., Du, N., and Bengio, S. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065*, 2017.

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pp. 6573–6583, 2017.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, pp. 6754–6764, 2017.

Rasmussen, J. G. Temporal point processes:the conditional intensity function, 2011. URL http://people.math.aau.dk/{~}jgr/teaching/punktproc11/tpp.pdf.

Rodriguez, M. G., Balduzzi, D., and Schölkopf, B. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011.

Sharma, A., Johnson, R., Engert, F., and Linderman, S. Point process latent variable models of larval zebrafish behavior. In *Advances in Neural Information Processing Systems*, pp. 10919–10930, 2018.

Wang, Y., Liu, S., Shen, H., Gao, J., and Cheng, X. Marked temporal dynamics modeling based on recurrent neural network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 786–798. Springer, 2017.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 2005. ISSN 0885-6125. 10.1007/bf00992696.

Xiao, S., Farajtabar, M., Ye, X., Yan, J., Song, L., and Zha, H. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, pp. 3247–3257, 2017a.

Xiao, S., Yan, J., Farajtabar, M., Song, L., Yang, X., and Zha, H. Joint modeling of event sequence and time series with attentional twin recurrent neural networks. *arXiv preprint arXiv:1703.08524*, 2017b.

Xiao, S., Yan, J., Yang, X., Zha, H., and Chu, S. M. Modeling the intensity function of point process via recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017c.

Xiao, S., Xu, H., Yan, J., Farajtabar, M., Yang, X., Song, L., and Zha, H. Learning conditional generative models for temporal point processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xu, H. and Zha, H. A dirichlet mixture model of hawkes processes for event sequence clustering. In *Advances in Neural Information Processing Systems*, pp. 1354–1363, 2017.

Yang, G., Cai, Y., and Reddy, C. K. Recurrent spatio-temporal point process for check-in time prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2203–2211. ACM, 2018.

Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. 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*, pp. 1513–1522. ACM, 2015.

Zhao, S., Song, J., and Ermon, S. Learning hierarchical features from deep generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 4091–4099. JMLR. org, 2017.

A. Related Work

Temporal point processes have been widely used for modeling complex dynamics in the field of finance, econometrics, information diffusion (Du et al., 2013; Rodriguez et al., 2011; Zhao et al., 2015). Hawkes process has been used to model self-exciting events such as after-shocks of earthquakes (Hawkes & Oakes, 1974; Bacry et al., 2016). However, most standard point processes incorporate some assumptions about the distribution of the time intervals. In the case of the Poisson process, distribution of event intervals lengths is a stationary process while non-homogenous Poisson process model time-dependent intervals, while still being independent of the history. Hawkes processes and self-correcting processes have explicit dependence on history to model future time steps.

In a different formulation, Du et al. (2016) use a recurrent neural network to model marked temporal point processes. The RNN models do not need any specific assumption about the model, and yet are very effective in learning future timesteps from history through their memory state. Furthermore, recently different neural network structure has been proposed to model either the conditional intensity function or explicit time interval successfully (Yang et al., 2018; Xiao et al., 2017a;0; Zhao et al., 2017; Li et al., 2017; Wang et al., 2017; Xiao et al., 2017b). The use of a neural network to compute the Hawkes process parameter has been also proposed (Mei & Eisner, 2017).

On the other hand, models have been proposed to learn a hierarchical structure of the point process. Dirichlet Hawkes process has been proposed to learn mixture of Hawkes process in the presence of multiple clusters (Xu & Zha, 2017). In (Sharma et al., 2018), the authors proposed a sequential state space model to understand larval zebrafish behavior. Our work takes the latter approach where we like to learn some disentangled representation. While RNN is powerful in modeling time-intensity function, explanability is often lost due to their black-box nature. On the other hand, parameterizing point processes using the standard model may not be enough. Real world data might be a mixture of different standard processes with different parameters.

There has been a lot of progress in stochastic sequential modeling. It has been shown that deterministic hidden states are not enough to model real-world stochasticity (Fraccaro et al., 2016; Krishnan et al., 2017). Based on the seminal work of Kingma & Welling (2013) on auto-encoding variational Bayes, several stochastic sequential models have been proposed, showing state of the art log-likelihoods on speech and audio datasets (Chung et al., 2015; Bayer & Osendorfer, 2014; Maddison et al., 2017; Goyal et al., 2017). However, learning a hierarchical structure from a deep generative model has not found much success (Zhao et al., 2017). It has been observed that if the lower layer latent state can max-

imize the ELBO objective, a higher layer latent state will collapse to prior distribution minimizing the KL divergence loss. Several tricks have been applied to learn to prevent this. In the KL annealing trick, loss from KL divergence is given weight from 0 to 1 over several epochs (Fraccaro et al., 2016). In β-VAE, authors outweighed KL loss to learn disentangled useful representation (Higgins et al., 2017). Discrete latent variables pose their own challenge in that they're not amenable to the reparameterization trick. Thus, several algorithms have been proposed using biased Gumbel softmax trick, unbiased REINFORCE algorithm (Jang et al., 2016; Williams, 2005). While using Gumbel-softmax distribution to approximate categorical distribution results in a biased estimate, variance decreases largely compared to the REINFORCE estimator.

B. Model Derivation

In this section, we use $\vec{\mathbf{a}}$ as a shorthand for $\mathbf{a}_{1:T}$. We can derive the lower bound on the likelihood $\mathcal{L}\left((\mathbf{x}, \mathbf{t})_{1:T}\right)$:

$$\begin{aligned}
&\log p_\theta((\mathbf{x}, \mathbf{t})_{1:T}) \\
&\quad \geq \int_{\mathbf{y},\vec{\mathbf{z}},\vec{\mathbf{h}}} q_\phi(\vec{\mathbf{z}}, \mathbf{y}, \vec{\mathbf{h}} \mid \vec{\mathbf{x}}, \vec{\mathbf{t}}) \log \frac{p_\theta(\vec{\mathbf{x}}, \vec{\mathbf{t}}, \mathbf{y}, \vec{\mathbf{z}}, \vec{\mathbf{h}})}{q_\phi(\vec{\mathbf{z}}, \mathbf{y}, \vec{\mathbf{h}} \mid \vec{\mathbf{x}}, \vec{\mathbf{t}})} \\
&\qquad = \mathbb{E}_{q_\phi(\vec{\mathbf{z}},\mathbf{y}\mid\vec{\mathbf{x}},\vec{\mathbf{t}})}\left[p_\theta(\vec{\mathbf{x}}, \vec{\mathbf{t}} \mid \mathbf{y}, \vec{\mathbf{z}}, \vec{\mathbf{h}})\right] \\
&\qquad\qquad - \mathrm{KL}\left(q_\phi(\vec{\mathbf{z}}, \mathbf{y} \mid \vec{\mathbf{x}}, \vec{\mathbf{t}}) \mid\mid p_\theta(\mathbf{y}, \vec{\mathbf{z}})\right)
\end{aligned}$$

For simplicity of the equations, we replace $(\mathbf{x}, \mathbf{t})$, with $\mathbf{x}$.

B.1. Model 1

The expectation term can be written as:

$$\mathbb{E}_{q_\phi(\mathbf{y},\vec{\mathbf{z}},\vec{\mathbf{h}}\mid\vec{\mathbf{x}})}\left[\prod_{t=1}^{T} \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \mathbf{h}_t, \mathbf{z}_t)\right]$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \int_{\vec{\mathbf{h}}} \int_{\mathbf{y}} \int_{\vec{\mathbf{z}}} q_\phi(\vec{\mathbf{h}} \mid \vec{\mathbf{x}}) q_\phi(\mathbf{y} \mid \vec{\mathbf{h}}, \vec{\mathbf{x}}) \\
&\qquad \times q_\phi(\vec{\mathbf{z}} \mid \vec{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}}) \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \mathbf{h}_t, \mathbf{z}_t)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \int_{\mathbf{y}} \int_{\mathbf{z}_t} q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_t, \mathbf{y}, \mathbf{x}_t) \\
&\qquad \times \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \tilde{\mathbf{h}}_t, \mathbf{z}_t)
\end{aligned}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{y}\mid\tilde{\mathbf{h}},\vec{\mathbf{x}})}\, \mathbb{E}_{q_\phi(\mathbf{z}_t\mid\tilde{\mathbf{h}}_t,\mathbf{y},\mathbf{x}_t)}\left[\log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \tilde{\mathbf{h}}_t, \mathbf{z}_t)\right]$$

The KL term:

$$\mathrm{KL}\left(q_\phi(\mathbf{y}, \vec{\mathbf{z}}, \vec{\mathbf{h}} \mid \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y}) \prod_{t=1}^{T} p_\theta(\mathbf{h}_t|\mathbf{h}_{t-1}\mathbf{x}_{t-1}) p_\theta(\mathbf{z}_t)\right)$$

$$\begin{aligned}
&= \int_{\mathbf{y}} \int_{\vec{\mathbf{z}}} q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) q_\phi(\vec{\mathbf{z}} \mid \tilde{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}}) \\
&\qquad \times \left[\log \frac{q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}})}{p_\theta(\mathbf{y})} + \log \frac{q_\phi(\vec{\mathbf{z}} \mid \tilde{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}})}{\prod_{t=1}^{T} p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1})}\right]
\end{aligned}$$

$$\begin{aligned}
&= \mathrm{KL}\left(q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y})\right) + \sum_{t=1}^{T} \int_{\mathbf{y}} \int_{\mathbf{z}_t} q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) \\
&\qquad \times q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_t, \mathbf{y}, \mathbf{x}_t) \log \frac{q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_t, \mathbf{y}, \mathbf{x}_t)}{p_\theta(\mathbf{z}_t)}
\end{aligned}$$

$$\begin{aligned}
&= \mathrm{KL}\left(q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y})\right) \\
&\quad + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{y}\mid\tilde{\mathbf{h}},\vec{\mathbf{x}})}\, \mathrm{KL}\left(q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_t, \mathbf{y}, \mathbf{x}_t) \mid\mid p_\theta(\mathbf{z}_t)\right)
\end{aligned}$$

B.2. Model 2

The expectation term:

$$\mathbb{E}_{q_\phi(\mathbf{y},\vec{\mathbf{z}},\vec{\mathbf{h}}\mid\vec{\mathbf{x}})}\left[\prod_{t=1}^{T} \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \mathbf{h}_t, \mathbf{z}_t)\right]$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \int_{\vec{\mathbf{h}}} \int_{\mathbf{y}} \int_{\vec{\mathbf{z}}} q_\phi(\vec{\mathbf{h}} \mid \vec{\mathbf{x}}) q_\phi(\mathbf{y} \mid \vec{\mathbf{h}}, \vec{\mathbf{x}}) \\
&\qquad \times q_\phi(\vec{\mathbf{z}} \mid \vec{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}}) \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \mathbf{h}_t, \mathbf{z}_t)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \int_{\mathbf{y}} \int_{\mathbf{z}_t} q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_{t:T}, \mathbf{y}, \mathbf{x}_{t:T}, \mathbf{z}_{t-1}) \\
&\qquad \times \log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \tilde{\mathbf{h}}_t, \mathbf{z}_t)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{y}\mid\tilde{\mathbf{h}},\vec{\mathbf{x}})}\, \mathbb{E}_{q_\phi(\mathbf{z}_t\mid\tilde{\mathbf{h}}_{t:T},\mathbf{y},\mathbf{x}_{t:T},\mathbf{z}_{t-1})} \\
&\qquad \left[\log p_\theta(\mathbf{x}_t \mid \mathbf{y}, \tilde{\mathbf{h}}_t, \mathbf{z}_t)\right]
\end{aligned}$$

The KL term:

$$\mathrm{KL}\left(q_\phi(\mathbf{y}, \vec{\mathbf{z}}, \vec{\mathbf{h}} \mid \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y}) \prod_{t=1}^{T} p_\theta(\mathbf{h}_t|\mathbf{h}_{t-1}\mathbf{x}_{t-1}) p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1})\right)$$

$$= \int_{\mathbf{y}} \int_{\vec{\mathbf{z}}} q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) q_\phi(\vec{\mathbf{z}} \mid \tilde{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}}) \times \left[\log \frac{q_\phi(\vec{\mathbf{z}} \mid \tilde{\mathbf{h}}, \mathbf{y}, \vec{\mathbf{x}})}{\prod_{t=1}^{T} p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1})} + \log \frac{q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}})}{p_\theta(\mathbf{y})} \right]$$

$$= \mathrm{KL}\left(q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y}) \right) + \sum_{t=1}^{T} \left[\int_{\mathbf{y}, \mathbf{z}_{t-1}, \mathbf{z}_t} q_\phi(\mathbf{z}_{t-1} \mid \tilde{\mathbf{h}}_{t-1:T}, \mathbf{y}, \mathbf{x}_{t-1:T}, \mathbf{z}_{t-2}) \times q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_{t:T}, \mathbf{y}, \mathbf{x}_{t:T}, \mathbf{z}_{t-1}) \times \log \frac{q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_{t:T}, \mathbf{y}, \mathbf{x}_{t:T}, \mathbf{z}_{t-1})}{p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1})} \right]$$

$$= \mathrm{KL}\left(q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}}) \mid\mid p_\theta(\mathbf{y}) \right) + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{y} \mid \tilde{\mathbf{h}}, \vec{\mathbf{x}})} \mathbb{E}_{q_\phi(\mathbf{z}_{t-1} \mid \tilde{\mathbf{h}}_{t-1:T}, \mathbf{y}, \mathbf{x}_{t-1:T}, \mathbf{z}_{t-2})} \left[\mathrm{KL}\left(q_\phi(\mathbf{z}_t \mid \tilde{\mathbf{h}}_{t:T}, \mathbf{y}, \mathbf{x}_{t:T}, \mathbf{z}_{t-1}) \mid\mid p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \right) \right]$$

C. More Experimental Details

C.1. Synthetic Dataset

To illustrate the effect of mixture of multiple disparate point processes, we generate sequences from Hawkes process, Homogenous Poisson Process, Non-homogenous Poisson process and Self-correcting process.

C.1.1. Hawkes process

Conditional Intensity function of Hawkes process is defined as:

$$\lambda^*(\mathbf{t}) = \lambda_0 + \alpha \sum_{i:\mathbf{t}_i < \mathbf{t}} \exp\left(-\frac{\mathbf{t} - \mathbf{t}_i}{\sigma} \right)$$

The parameters necessary for defining this density are $\Theta = \{\lambda_0, \alpha, \beta\}$. We sample $\lambda_0 \sim \text{unif}(0.2, 1), \alpha \sim \text{unif}(0, 1), \sigma \sim \text{unif}(\alpha, 1)$. We do this k times, where $k \in \{4, 8, 16\}$. For each of the values of k

C.1.2. Self-correcting process

For self-correcting point process, conditional intensity function can be written as:

$$\lambda^*(\mathbf{t}) = \exp(\mu \mathbf{t} - \sum_{i:\mathbf{t}_i < \mathbf{t}} \alpha)$$

The parameters necessary for defining this density are $\Theta = \{\mu, \alpha\}$. We sample $\mu \sim \text{unif}(0.5, 1.5), \alpha \sim \text{unif}(0.1, .5)$.

C.1.3. Homogeneous Poisson process

For Homogeneous Poisson process, the intensity function is constant, i.e., $\lambda^*(\mathbf{t}) = \lambda_0$. The parameters necessary for defining this density are $\Theta = \{\lambda_0\}$, sampled as $\alpha \sim \text{unif}(0.1, 1)$.

C.1.4. Non-homogeneous Poisson process

For non-homogeneous Poisson process, the intensity function is independent of the past, but is still a function of time. We define the intensity function as:

$$\lambda^*(\mathbf{t}) = 0.08 \cdot (0.1\sin(\pi \mathbf{t}) + 0.3\cos(0.4\pi \mathbf{t}) + 1)$$

For the purposes of illustration, we did not incorporate any marker information into the sequences.

C.2. Real-world Dataset

We provided empirical performance on six real world datasets from various domain used in previous research (Du et al., 2016; Mei & Eisner, 2017). For all datasets, 20% of training data was further splitted in validation dataset.

- **MIMIC-II** contains deidentified clinical visits information from critical care patient from the year 2002 to 2009. Event data encodes major disease of the patients over time and the time information represents the interval after which patient visits the hospital. We used benchmark datasets from earlier research [3]. There are 585 and 65 patients in the training and testing datasets. Number of disease was 75.
- **Financial Transaction dataset** was collected from raw limited book order of high frequency transaction from NYSE for a stock. Events are either Buy or Sell. Input data contains a single sequence of buy or sell events. Train and test sequence length is 624,149 and 69,350 [3].
- **Stackoverflow Dataset** contains awarded badge information of the users. The data was collected from the period of 2012-01-01 to 2014-01-01 resulting $480K$ events from $6K$ users for a total of 22 different types of badges [3].
- **Meme-Tracker Dataset** has 5000 event or online media site. The sequence for a meme defines what time it will be mentioned at some website. [4]Training dataset contains 93267 event tokens and test dataset contains 14932 event tokes.
- **Retweet Dataset** contains retweet history of many tweet sequences. Event type classifies popularity of

[3] https://github.com/dunan/NeuralPointProcess
[4] https://github.com/HMEIatJHU/neurawkes

retweeter among small, medium and large tweeter. [4] Number of training and testing events are 1739547 and 215521.

- **LastFM** dataset contains sequential listening habits of users involving millions of songs. The datasets contains 3150 events [3].

C.3. Experimental Framework

For all experiments, we used early stopping on validation dataset, and the model at stopping epoch was used for testing. We only computed log-likelihood for the models.

C.3.1. HYPER-PARAMETERS

We used the following hyper-parameter settings:

1. We used GRU implementation of RNN in PyTorch. Hidden state size of the RNN is varied from $\{128, 256, 512\}$.

2. Batch size was fixed to 32 for all datasets, except for financial transaction dataset which had only few long sequences as train data. We used batch size of 10 for financial transaction dataset.

3. We used Maxgradnorm, l2 as well as dropout regularizations. Maxgradnorm was varied in $\{0.1, 1., 10.\}$. Dropout was varied in $\{0.25, 0.4\}$. L2 (weight decay) regularization was varied in $\{10^{-5}, 10^{-3}\}$.

4. Number of cluster (k for variable $\mathbf{y}$) was set to 10 for Model 1 and Model 2. Latent variable $\mathbf{z}$ dimension was fixed to 32.

5. We annealed KL loss from 0 to 1 over 40 epochs.

6. Learning rates that were tried were $1e-3$ and $1e-4$. The optimizer used was ADAM.

C.3.2. NETWORK ARCHITECTURE

The network architecture was fixed as follows.

1. The marker and time information was embedded into fixed dimension of 128 and 8 respectively before feeding to RNN. We used a shared representation layer of dimension 256 from hidden states to output before creating the output event and time prediction.

2. We used ReLU as a non-linearity after each hidden state. From shared output representation, we used two multi-layer perceptron to predict time as well as event outputs. We used dropout regularizer at this layer.

3. For all models, we compute $\mu_{\mathbf{t}}$ as well as $\log \text{var}_{\mathbf{t}}$ instead of a point estimation for the next event's timestamp. This allows us to compute log-likelihood while flexibly modeling variance in the observation.

4. For modeling marker information, the marker network models the logits in case of categorical marker. For real valued marker, similar to time, we parameterize mean and standard deviation of the output using a Normal distribution. For multi-label marker, we used a sigmoid at the output layer to separate give us the probability of each of the events occurring.

5. For our proposed model, we used additional reverse RNN which takes input as shown in Figure 1. Hidden dimension of the reverse RNN is kept same as the forward RNN.

6. The inference module for $\mathbf{z}$ is shared over time and implemented using a MLP with a ReLU non-linearity and dropout regularization. The inference network for Categorical random variable was parameterized using the final state of the Reverse RNN.

7. The full generative and inference network was then trained with stochastic gradient ascent with maxgradnorm and weight decay.