Appendix

Qiang Zhang [1] **Aldo Lipani** [1] **Omer Kirnap** [1] **Emine Yilmaz** [1 2]

A. Optimization

To learn the parameters of the proposed method, we perform a Maximum Likelihood Estimation (MLE). Other advanced and more complex adversarial learning (Xiao et al., 2017) and reinforcement learning (Li et al., 2018) methods have been proposed, however we use MLE for its simplicity. We use the same optimization method for our model and all baselines as done in their original papers. To apply MLE, we derive a loss function based on the negative log-likelihood.

Given the history $\mathcal{H}_t$ and the next timestamp t_i, the probability of v_i is calculated as

$$p(u = v_i|t_i) = \frac{\lambda_{v_i}(t_i)}{\lambda(t_i)}, \tag{1}$$

where $\lambda(t) = \sum_{u=1}^{u=|\mathcal{U}|} \lambda_u(t)$. The log-likelihood of an event sequence $\mathcal{S}$ over a time interval $[0, T]$ is given by:

$$\begin{aligned}
\mathcal{L} &= \log\left[\prod_{i=1}^{L} p(t_i)p(u = v_i|t_i)\right] \\
&= \log\left[\prod_{i=1}^{L} \lambda(t_i)\exp\left(-\int_{t_{i-1}}^{t_i}\lambda(\tau)\mathrm{d}\tau\right)\frac{\lambda_{v_i}(t_i)}{\lambda(t_i)}\right] \\
&= \log\left[\exp\left(-\sum_{i=1}^{L}\int_{t_{i-1}}^{t_i}\lambda(\tau)\mathrm{d}\tau\right)\prod_{i=1}^{L}\lambda_{v_i}(t_i)\right] \\
&= \log\left[\exp\left(-\int_{0}^{T}\lambda(\tau)\mathrm{d}\tau\right)\prod_{i=1}^{L}\lambda_{v_i}(t_i)\right] \\
&= \sum_{i=1}^{L}\log\lambda_{v_i}(t_i) - \int_{0}^{T}\lambda(\tau)d\tau
\end{aligned} \tag{2}$$

where the first term is the sum of the log-intensity functions of past events, and the second term corresponds to the log-likelihood of infinitely many non-events. By non-event, we mean there are no events at a certain timestamp. Intuitively, the probability that there is no event of any type in the infinitesimally time interval $[t, t+dt)$ is equal to $1-\lambda(t)dt$, the log of which is $-\lambda(t)dt$.

To maximize the log-likelihood objective with stochastic gradient methods, we need to get an unbiased estimate of the gradient with respect to the model parameters. Monte Carlo sampling is used to estimate the gradients of the integral part in Equation 2. It takes the average of several samples and thus reduces the variance of gradient estimation, which is also used in (Mei & Eisner, 2017).

B. Prediction

This section introduces how to predict future events including types and timestamps given a set of history events. According to Mei & Eisner (2017), given the history $\mathcal{H}(t_{i+1}) = \{(v_1, t_1), \ldots, (v_i, t_i)\}$, the time density of the subsequent event is calculated as:

$$\begin{aligned}
p_{i+1}(t) &= P(t_{i+1} = t|\mathcal{H}(t_{i+1})) \\
&= \lambda(t)\exp\left(-\int_{t_i}^{t}\lambda(s)ds\right).
\end{aligned} \tag{3}$$

The prediction of the next event timestamp t_{i+1} is equal to the following expectation:

$$\hat{t}_{i+1} = \mathbb{E}[t_{i+1}|\mathcal{H}(t_{i+1})] = \int_{t_i}^{\infty} t \times p_{t_{i+1}}(t)dt. \tag{4}$$

While the prediction of the event type is equal to:

$$\hat{u}_{i+1} = \arg\max_{u\in\mathcal{U}} \int_{t_i}^{\infty} \frac{\lambda_u(t)}{\lambda(t)} p_{i+1}(t)dt. \tag{5}$$

Because this integral is not solvable analytically we approximate it via Monte Carlo sampling.

C. Datasets

Retweet (RT) The Retweet dataset contains a total number of 24,000 retweet sequences. In each sequence, an event is a tuple of the tweet type and time. There are $U = 3$ types: "small", "medium" and "large" retweeters. The "small" retweeters are those who have fewer than 120 followers, "medium" retweeters have more than 120 but fewer than 1,363 followers, and the rest are "large" retweeters. As for retweet time, the first event in each sequence is labeled with 0, the next events are labeled with reference to their time interval with respect to the first event in this sequence. The dataset contains information of when a post will be retweeted by which type of users.

*Equal contribution [1]University College London, London, United Kingdom [2]Amazon, London, United Kingdom. Correspondence to: Qiang Zhang <qiang.zhang.16@ucl.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

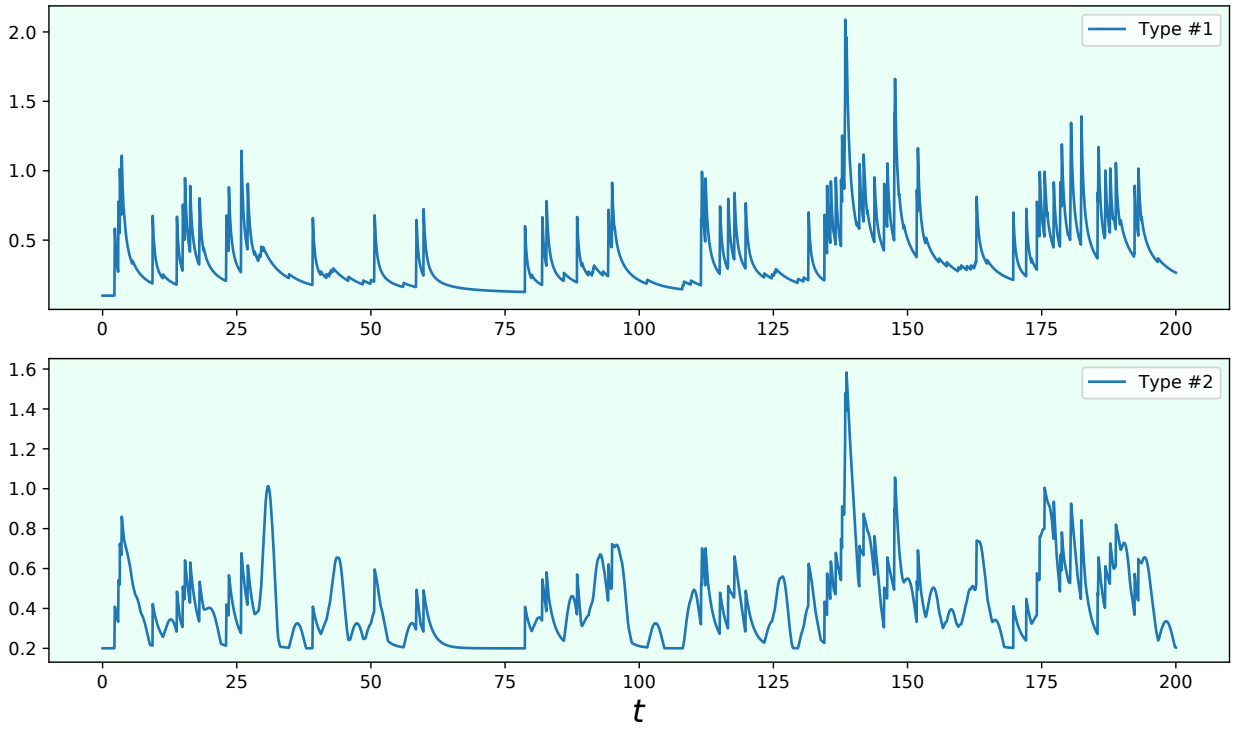

Figure 1. The intensities of the two-dimensional Hawkes processes over the synthetic dataset. The upper and the lower subfigure correspond to the dimension-1 and the dimension-2 respectively.

StackOverflow (SOF) The StackOverflow dataset includes sequences of user awards within two years. StackOverflow is a question-answering website where users are awarded based on their proposed questions and their answers to questions proposed by others. This dataset contains in total 6,633 sequences. There are $U = 22$ types of events: Nice Question, Good Answer, Guru, Popular Question, Famous Question, Nice Answer, Good Question, Caucus, Notable Question, Necromancer, Promoter, Yearling, Revival, Enlightened, Great Answer, Populist, Great Question, Constituent, Announcer, Stellar Question, Booster and Publicist. The award time records when a user receives an award. With this dataset, we can learn which type of awards will be given to a user and when.

MIMIC-II (MMC) The Multiparameter Intelligent Monitoring in Intensive Care (MIMIC-II) dataset is developed based on an electric medical record system. The dataset contains in total 650 sequences, each of which corresponds to an anonymous patient's clinical visits in a seven-year period. Each clinical event records the diagnosis result and the timestamp of that visit. The number of unique diagnosis results is $U = 75$. According to the clinical history, a point process captures the dynamics of when a patient will visit doctors and what the diagnose result will be.

References

Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., and Song, L. Learning temporal point processes via reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 10781–10791, 2018.

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.

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, 2017.