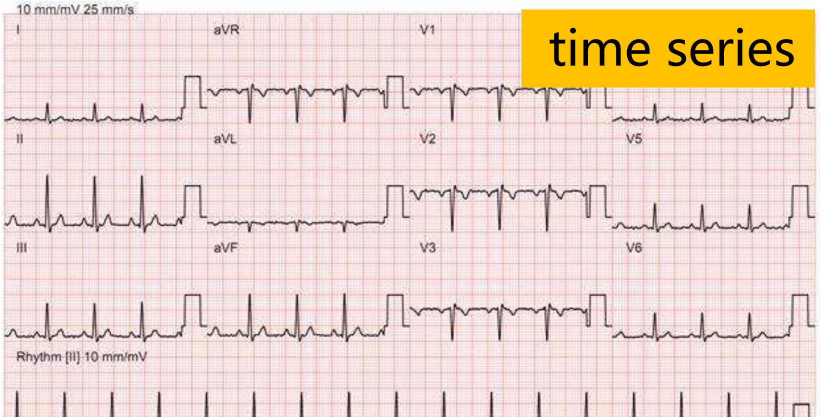
TEMPORAL POINT PROCESSES

JUNCHI YAN, THINKLAB@SJTU

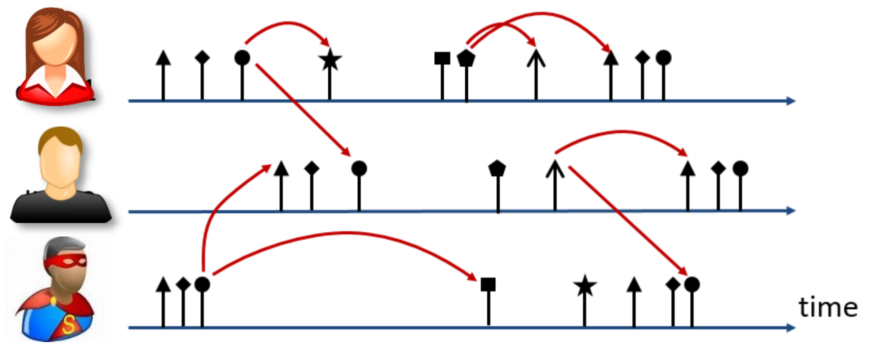
outline

1. Introduction of Point Process
2. Temporal Point Process: Basics
3. Deep Learning for Temporal Point Process

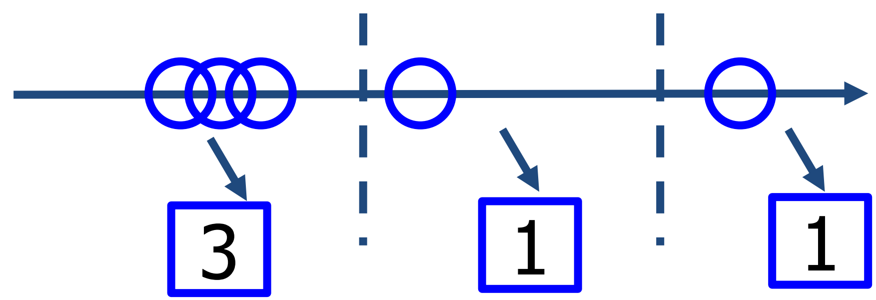
Event sequence vs other series



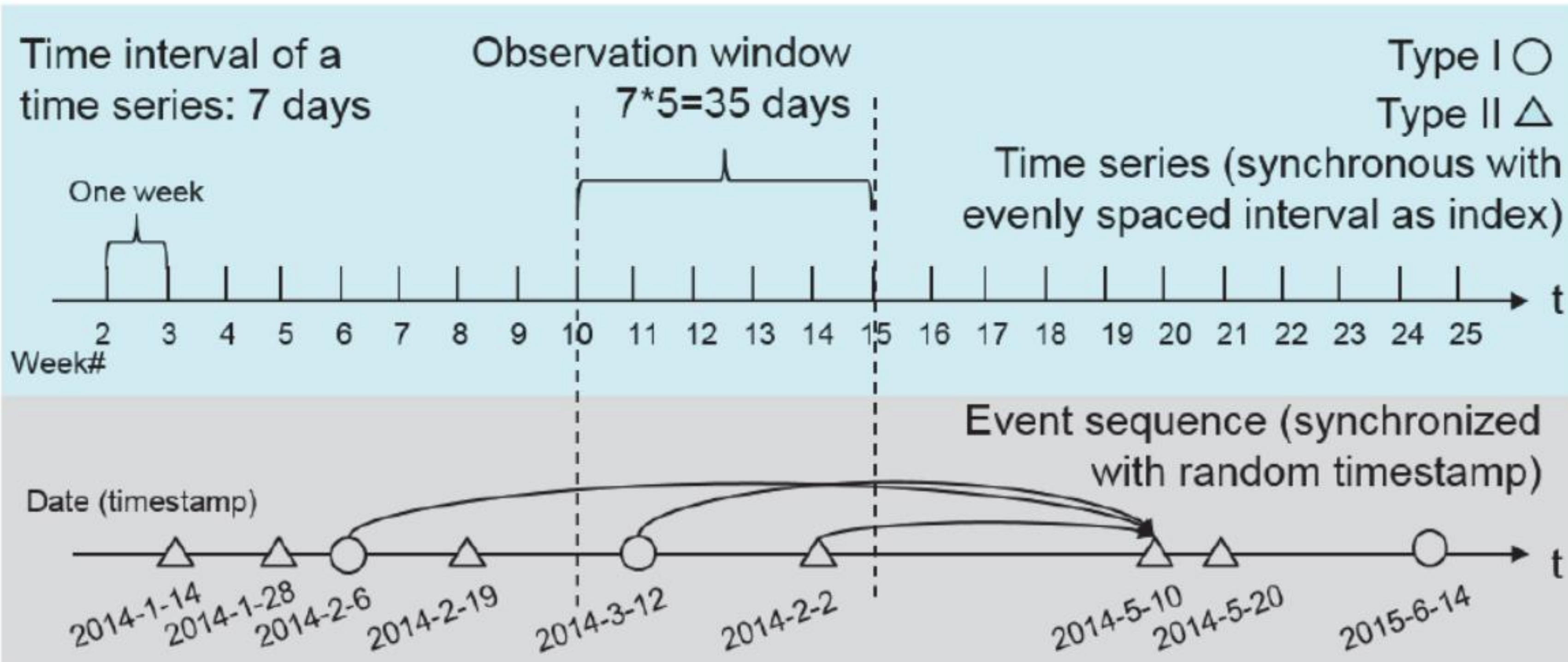
event sequence

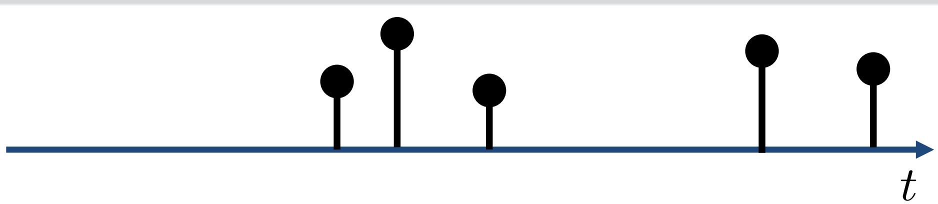


discretization



event data VS time series?





Discrete events in continuous time

Event data for point process

- **A variety of applications**

  — **infection and spread of contagious diseases (spatial info)**

  — **sequence of retweets in Tweeter (spatial info)**

  — **sequence of user queries submitted to a search engine (content info)**

  — **earthquakes with magnitudes with locations: spatial-temporal event modeling**

- **Goals of temporal event modeling**

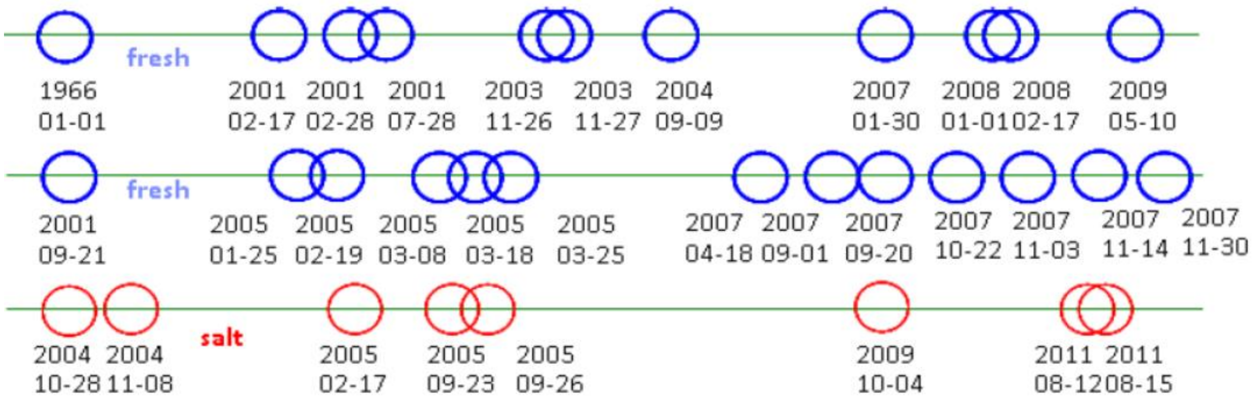
  — **studying the mechanisms that give rise to the dynamics of the recurrence of events**

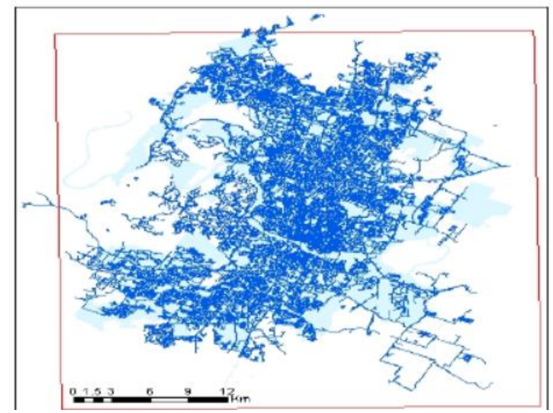
  — **predicting the dynamics of events in the future based on event history**

  — **designing intervention and control measures to steer the dynamics of events to desirable outcomes**

Underground Water Pipe Failure Data

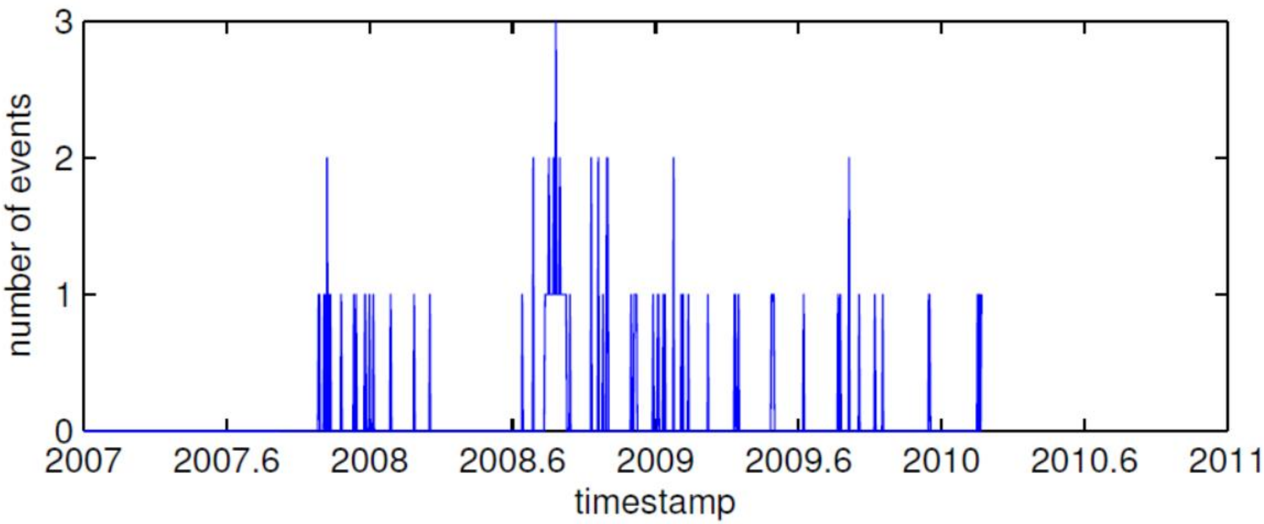
Management and maintenance of aging infrastructures

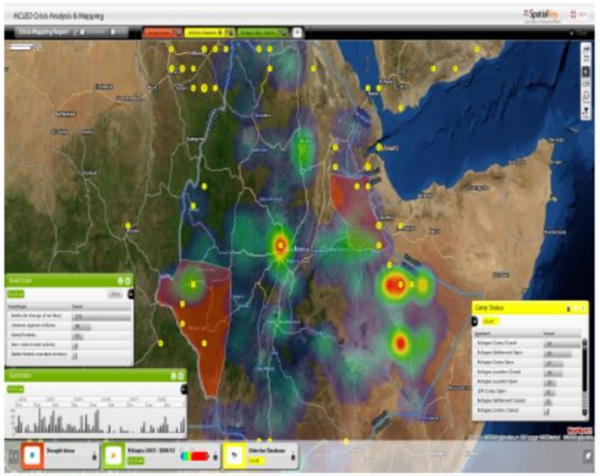




- **700K underground water pipes of a large Asian city: preventative rehabilitation and replacement are the key activities for pipe asset management**
- **Understanding of the failure mechanism in repairable pipes and modeling the stochastic behavior of the recurrences of pipe failure**

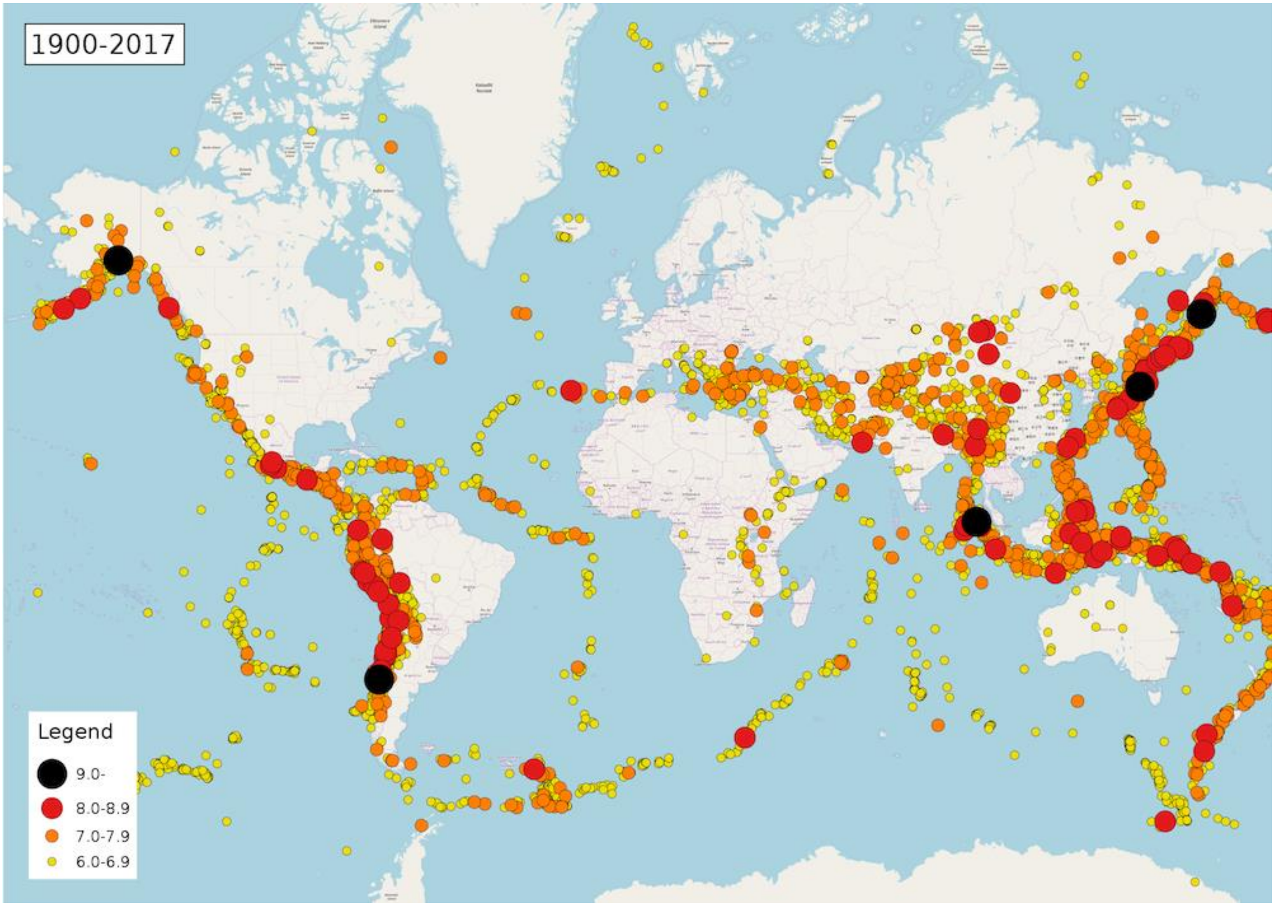
Armed Conflict Location and Event Data (ACLED)





- **36.3% dyadic events in the Afghan dataset are without the actor information**
- **an event with civilian casualty is observed but we did not observe who carried out the act**
- **Event attribution: infer the missing actors of events of which only the timestamps are known based on retaliation patterns** [Zammit, et al. 2012]

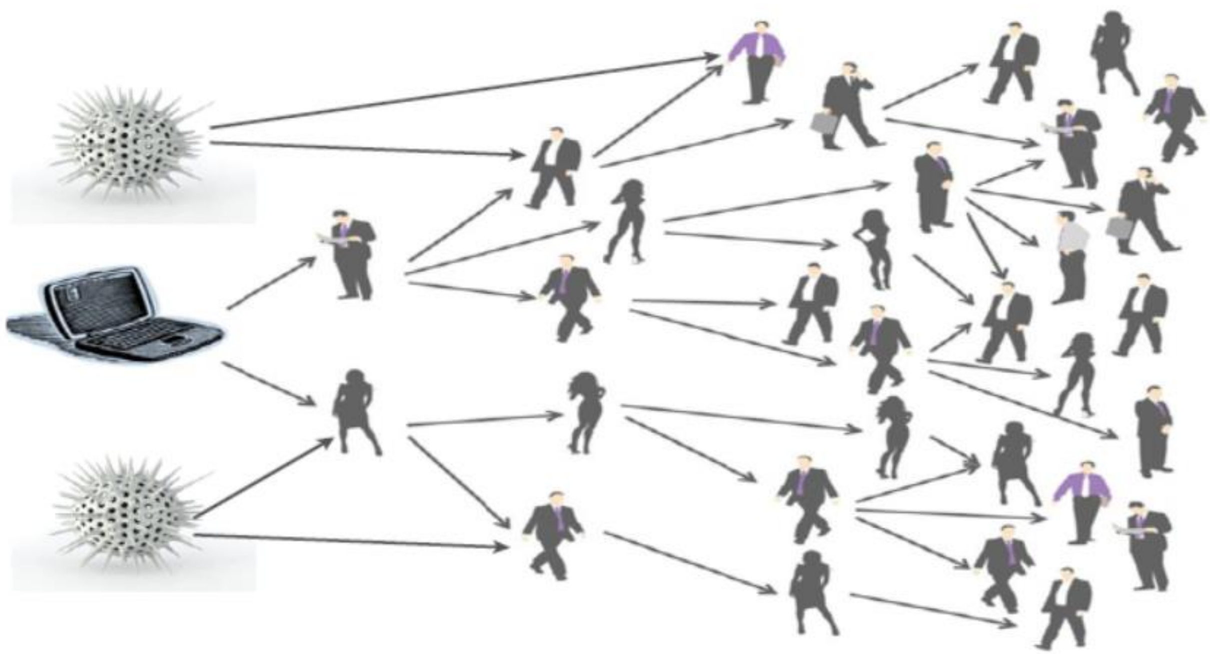
Space-time Point Process Models for Earthquake(ETAS)



Ogata, Y. (1988). Statistical models for earthquake occurrences and residual analysis for point processes. J. Amer. Statist. Assoc. 83 9–27.

Information propagation in Social Networks

- **Multiple memes are evolving and spreading through the same network**
- **Explore the content of the information diffusing through a network**
- **Simultaneous diffusion network inference and meme tracking**



[1] Learning parametric models for social infectivity in multidimensional Hawkes processes. In AAAI, 2014

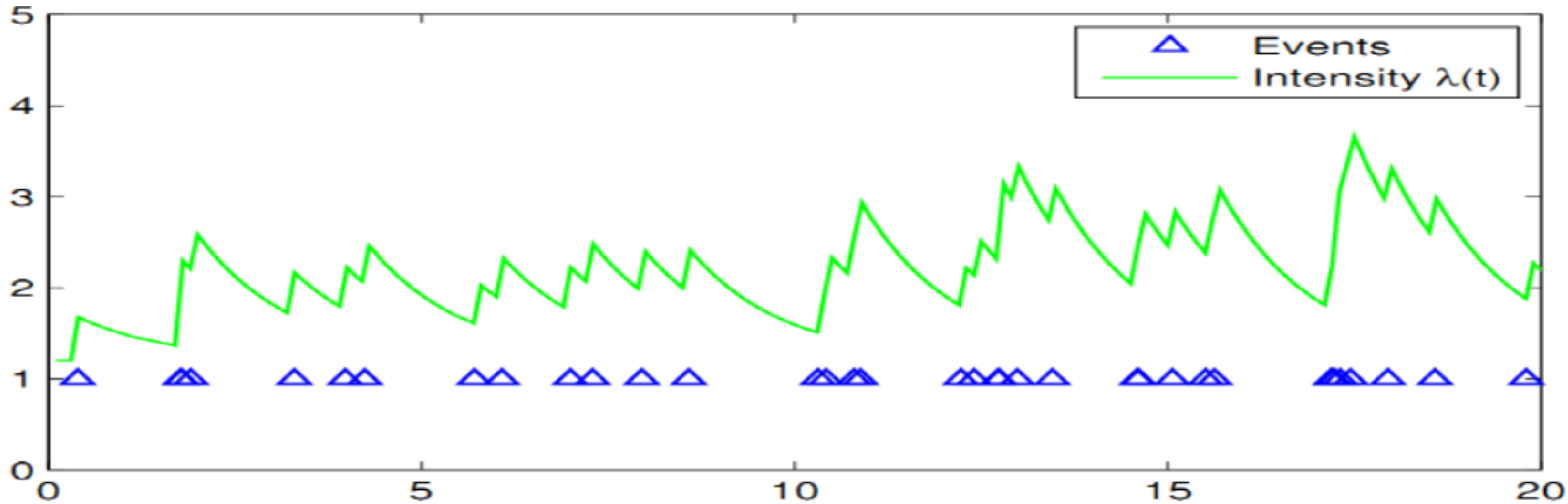
outline

1. Introduction of Point Process
2. Temporal Point Process: Basics
3. Deep Learning for Temporal Point Process

Event data for point process

- Conditional intensity

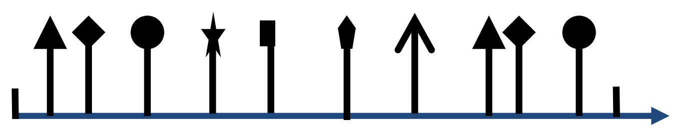
$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\mathbb{E}(\underbrace{\boxed{N(t+\Delta t)}}_{\text{event number}} - N(t) | \underbrace{\boxed{\mathcal{H}_t}}_{\text{history}})}{\Delta t} = \frac{\mathbb{E}(dN(t)|\mathcal{H}_t)}{dt}$$



Classical model of point process

Poisson processes:

$$\lambda^*(t) = \lambda$$

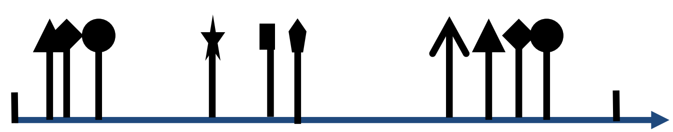


Terminating point processes:

$$\lambda^*(t) = g^*(t)(1 - N(t))$$

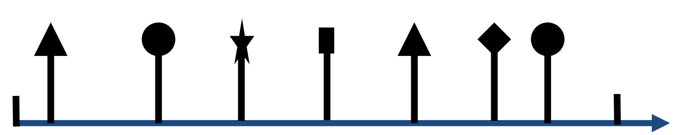
Self-exciting point processes:

$$\lambda^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i)$$



Self-correcting processes:

$$\lambda^*(t) = e^{\mu t - \sum_{t_i \in \mathcal{H}(t)} \alpha}$$



Conditional intensity function

- conditional intensity function (another definition):

mainly for next event

density function

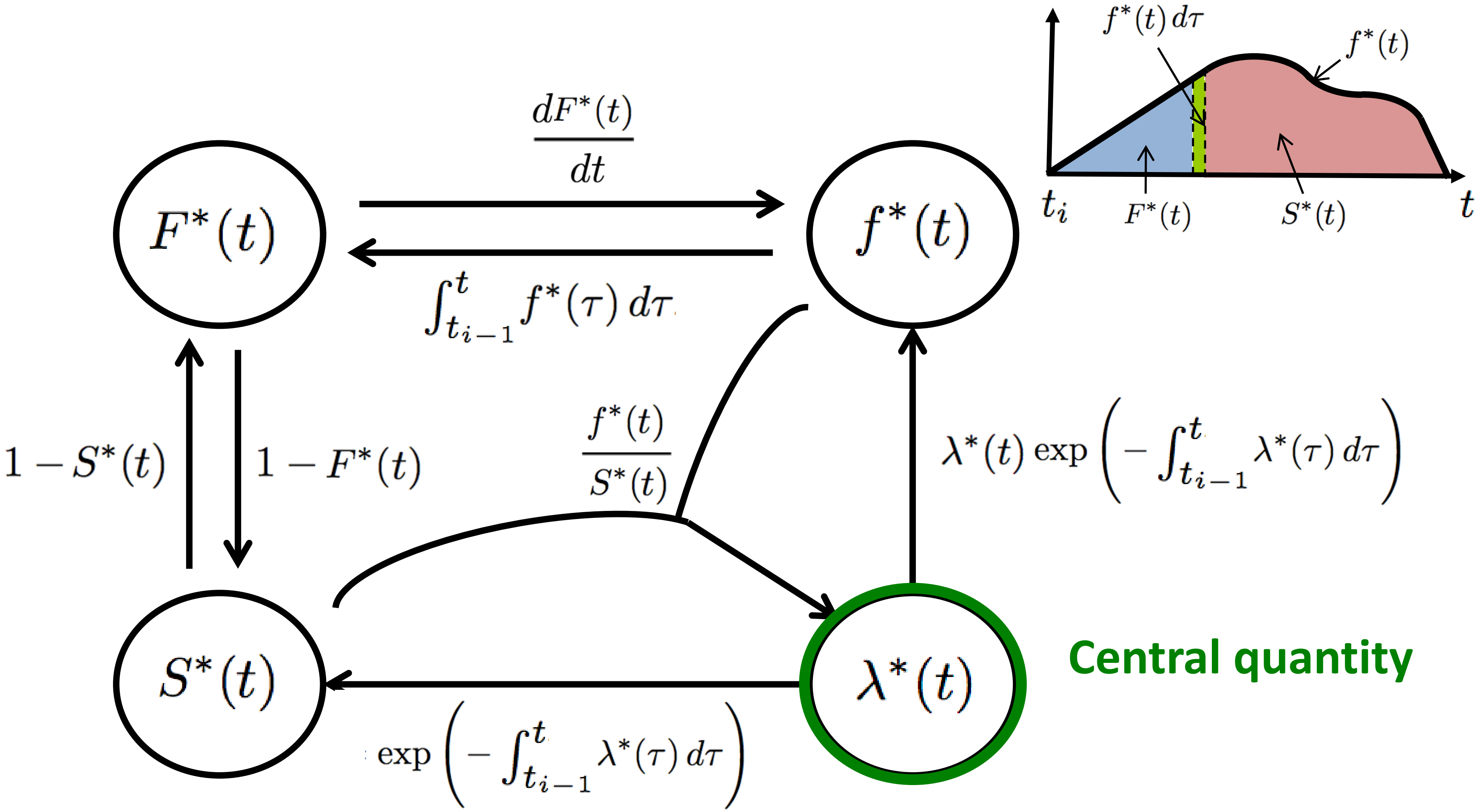
history

$$\lambda(t|\boxed{H}_{t_n}) = \frac{f(t|H_{t_n})}{1 - \boxed{F(t|H_{t_n})}} = \frac{f^*(t)}{1 - F^*(t)} = \frac{f^*(t)}{\boxed{S^*(t)}}$$

Cumulative distribution function

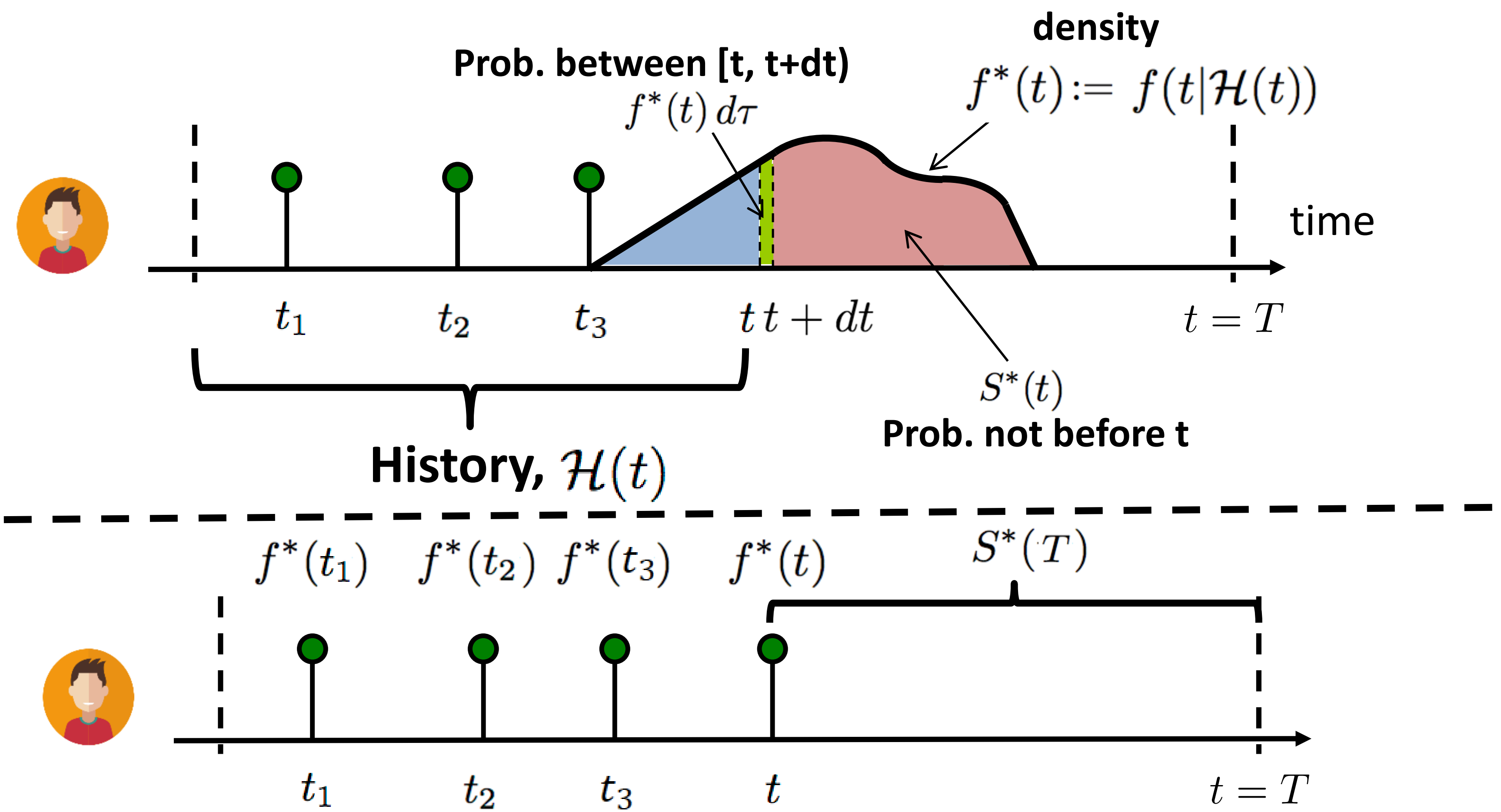
Survival function

Relation between f*, F*, S*, λ*



Central quantity

Density and likelihood



Likelihood of a timeline: $f^*(t_1)\ f^*(t_2)\ f^*(t_3)\ f^*(t)\ \ S^*(T)$

Likelihood

let $t_1 < t_2 < \cdots < t_{n-1} < t_n$, be the event times observed over $[0, T]$, use factorization, we can get the likelihood

$$L = f^*(t_1) \cdot f^*(t_2) \cdot \cdots \cdot f^*(t_n) \cdot S^*(T)$$
$$= \left(\prod_{i=1}^{n} \lambda^*(t_i) \cdot \exp\left(-\int_{t_{i-1}}^{t_i} \lambda^*(s) ds \right) \right) \cdot \exp\left(-\int_{t_n}^{T} \lambda^*(s) ds \right)$$
$$= \left(\prod_{i=1}^{n} \lambda^*(t_i) \right) \cdot \exp\left(-\int_{0}^{T} \lambda^*(s) ds \right)$$

Simulation—the inversed method

- Algorithm 1. The inverse method algorithm
- 1. set $t = 0$, $t_0 = 0$, $s_0 = 0$, $i = 1$
- 2. while $true$:
-   (i) generate $U \sim \mathrm{Uniform}([0,1])$
-   (ii) calculate $\tau_i = -(\log U)/\lambda$
-   (iii) set $s_i = s_{i-1} + \tau_i$
-   (iv) calculate t where $t = \Lambda^{*-1}(s_n)$
-   (v) if $t < T$: $i = i + 1, t_i = t$ else break
- Output: Retrieve the simulated process $\{t_n\}$ on $[0, T]$

Simulation—thinning method

- Algorithm 2. Ogata's modified thinning algorithm
- 1. set $t = 0$, $i = 1$
- 2. while $t \leq T$:
-   (i) calculate $m(t), l(t)$
-   (ii) generate $U \sim \mathrm{Unif}([0,1])$ then set $s = -(\log U)/\lambda$ and generate $U' \sim \mathrm{Unif}([0,1])$
-   (iii) if: $s > l(t)$, set $t = t + l(t)$
-   (iv) elif: $t + s > T$ or $U' > \lambda^*(t+s)/m(t)$, set $t = t + s$
-   (v) else: set $n = n + 1$, $t_n = t + s$, $t = t + s$
- Output: Retrieve the simulated process $\{t_n\}$ on $[0, T]$

Multi-dimensional Hawkes process

- Intensity of multi-dimensional Hawkes process: given event data $\{(t_i^m)_i\}_{m=1}^M$

$$\lambda_d = \mu_d + \sum_{i:t_i<t} \alpha_{dd_i} e^{-\beta(t-t_i)}$$

- where $\mu_d \geq 0$ is the base intensity for the d-th Hawkes process
- The coefficient α_{dd_i} captures the mutually exciting property between the d_i-th and the d-th dimension. It shows how much influence the events in d_i-th process have on future events in the d-th process.

Maximum-likelihood estimation

log-likelihood:

$$\log L = \sum_{d=1}^{M} \left\{ \sum_{(t_i,d_i)|d_i=d} \log \lambda_{d_i}(t_i) - \int_0^T \lambda_d(t)dt \right\}$$

$$= \sum_{i=1}^{n} \log\left(\mu_{d_i} + \sum_{t_j<t_i} \alpha_{d_i d_j} e^{-\beta(t_i-t_j)}\right) - T\sum_{d=1}^{M}\mu_d - \sum_{d=1}^{M}\sum_{j=1}^{n} \alpha_{dd_j} G_{dd_j}(T-t_j)$$

Jensen equality

$$\geq \sum_{i=1}^{n} \left(p_{ii} \log \frac{\mu_{d_i}}{p_{ii}} + \sum_{j=1}^{i-1} p_{ij} \log \frac{\alpha_{d_i d_j} e^{-\beta(t_i-t_j)}}{p_{ij}} \right) - T\sum_{d=1}^{M}\mu_d - \sum_{d=1}^{M}\sum_{j=1}^{n} \alpha_{dd_j} G_{dd_j}(T-t_j)$$

$$= Q(\theta|\theta^l)$$

EM algorithm

Maximum-likelihood estimation

- E-step

$$p_{ii}^{(k+1)} = \frac{\mu_{d_i}^{(k)}}{\mu_{d_i}^{(k)} + \sum_{j=1}^{i-1} \alpha_{d_i d_j}{}^{(k)} e^{-\beta(t_i - t_j)}}$$

$$p_{ij}^{(k+1)} = \frac{\alpha^{(k)} e^{-\beta(t_i - t_j)}}{\mu_{d_i}^{(k)} + \sum_{j=1}^{i-1} \alpha_{d_i d_j}{}^{(k)} e^{-\beta(t_i - t_j)}}$$

The probability that the event i is triggered by the base intensity μ

The probability that the event i is triggered by the event j

Maximum-likelihood estimation

- M-step (do partial differential equation for μ and α)

$$\mu_d^{(k+1)} = \frac{1}{T} \sum_{i=1, d_i=d}^{n} p_{ii}^{(k+1)}$$

$$\alpha_{uv}^{(k+1)} = \frac{\sum_{i=1,d_i=u}^{n} \sum_{j=1,d_j=v}^{i-1} p_{ij}^{(k+1)}}{\sum_{j=1,d_j=v}^{n} G\left(T - t_j\right)}$$

For β, if $e^{-\beta(T-t_i)} \approx 0$

$$\beta^{(k+1)} = \frac{\sum_{i>j} p_{ij}^{(k+1)}}{\sum_{i>j}(t_i - t_j) p_{ij}^{(k+1)}}$$

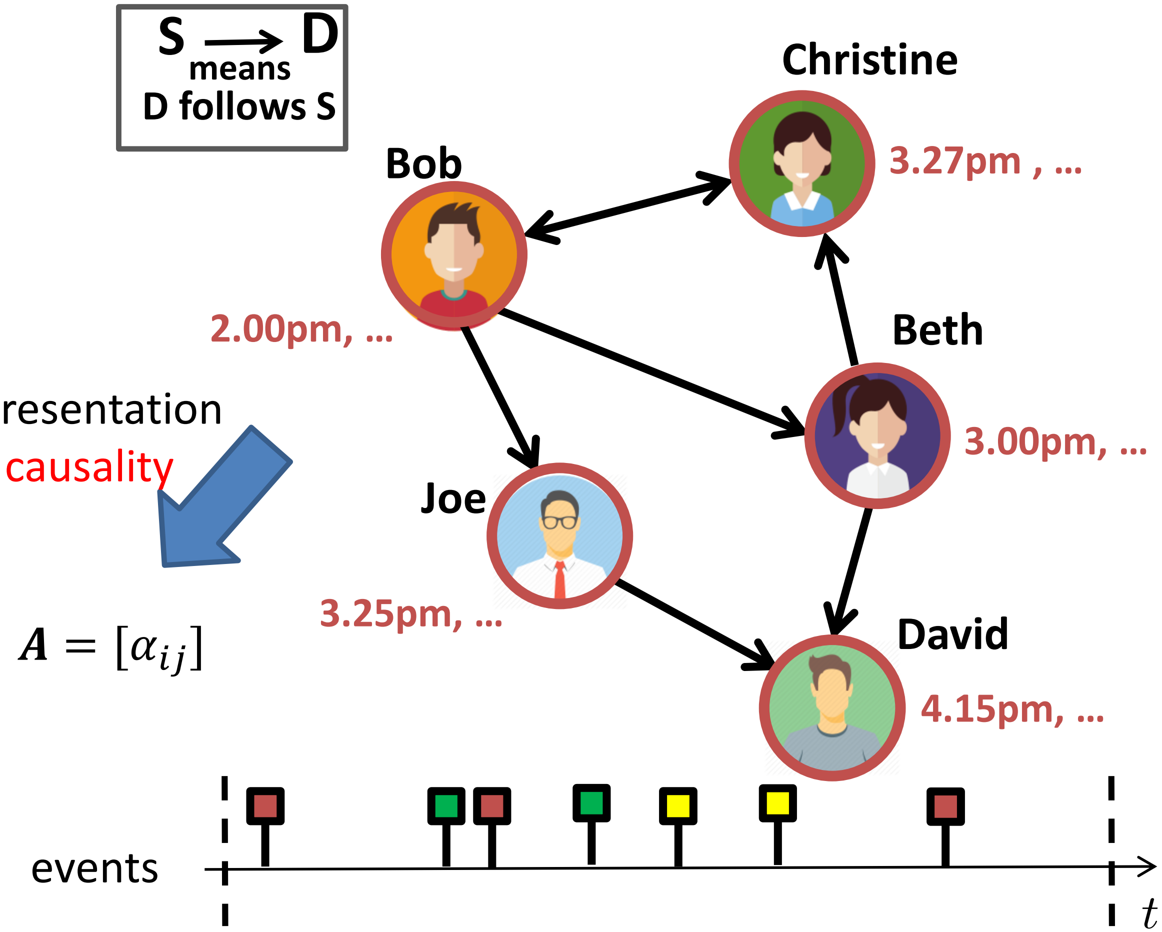
Applications

- For α_{ij}, influence from dimension i to j
- Social Infectivity
- If high dimension, overfitting for $\boldsymbol{A} = [\alpha_{ij}]$
- Sparse Low-rank Networks
- regularize the maximum likelihood estimator

$$\min_{A \geq 0, \mu \geq 0} -L(A, \mu) + \lambda_1 \|A\|_* + \lambda_2 \|A\|_1$$

- $\|A\|_*$ is the nuclear norm of matrix A, which is defined to be the sum of its singular value

Information propagation with MHP



Explicit representation of Granger causality

$$\boldsymbol{A} = [\alpha_{ij}]$$

outline

1. Introduction of Point Process
2. Temporal Point Process: Basics
3. Deep Learning for Temporal Point Process

Outline： Deep Learning for TPP

3.1 RNN model for TPP

3.2 Adversarial learning for TPP

3.3 Reinforcement learning for TPP

3.4 Embedding for multi-dimensional TPP

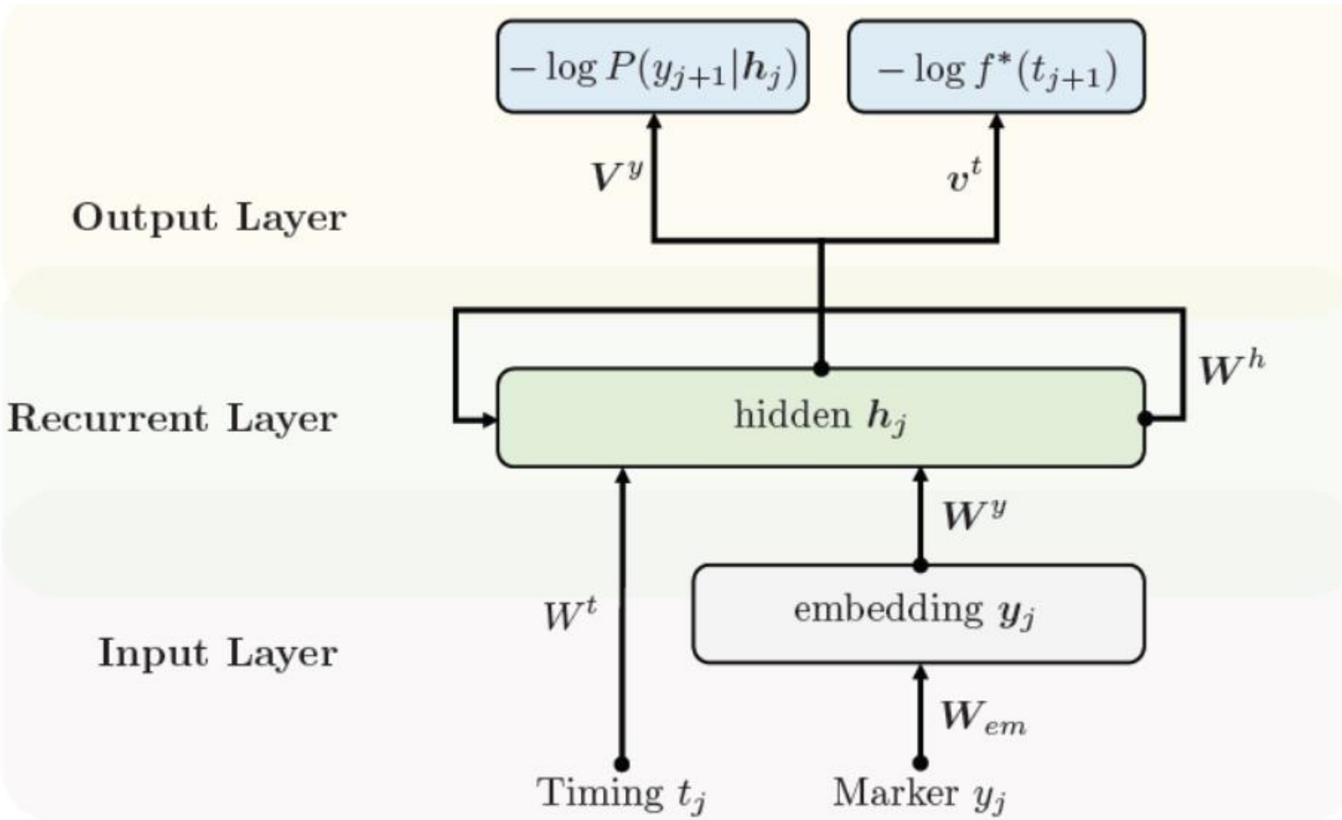
Deep point process

- In traditional way, you have to design the marked conditional intensity (or the temporal conditional intensity and density for the marked data) and find its tailored learning algorithm

- But in deep point process, it is much easier while using RNN (or LSTM and other deep model)

Deep point process

f*: conditional density function



[Nan et al 2016]

Architect

Recurrent marked temporal point processes: Embedding event history to vector. In KDD, 2016.

Deep point process

- For time prediction

  The simplest: Gaussian distribution(MSE)

$$f(t_{j+1}|h_j) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-\left(t_{j+1} - \tilde{t}_{j+1}\right)^2}{2\sigma^2}\right)$$

  $\tilde{t}_{j+1}$: the real timestamp of $j+1$

- Shortcoming1: can not get its analytical intensity
- Shortcoming2: the distribution of t_{j+1} has a probability that $t_{j+1} < t_j$

Deep point process

- So other conditional assumptions besides Gaussian distribution is OK such as exponential distribution.
- Method 2 for time prediction: intensity assumption

For example with intensity based on 3 term

$$\lambda^*(t) = \exp(\underbrace{\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j}_{\text{past influence}} + \underbrace{w(t - t_j)}_{\text{current influence}} + \underbrace{b}_{\text{base intensity}})$$

- The first term: represents the accumulative influence from the marker and the timing information of the past events

Deep point process

$$\lambda^*(t) = \exp(\underbrace{\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j}_{\text{past influence}} + \underbrace{w(t - t_j)}_{\text{current influence}} + \underbrace{b}_{\text{base intensity}})$$

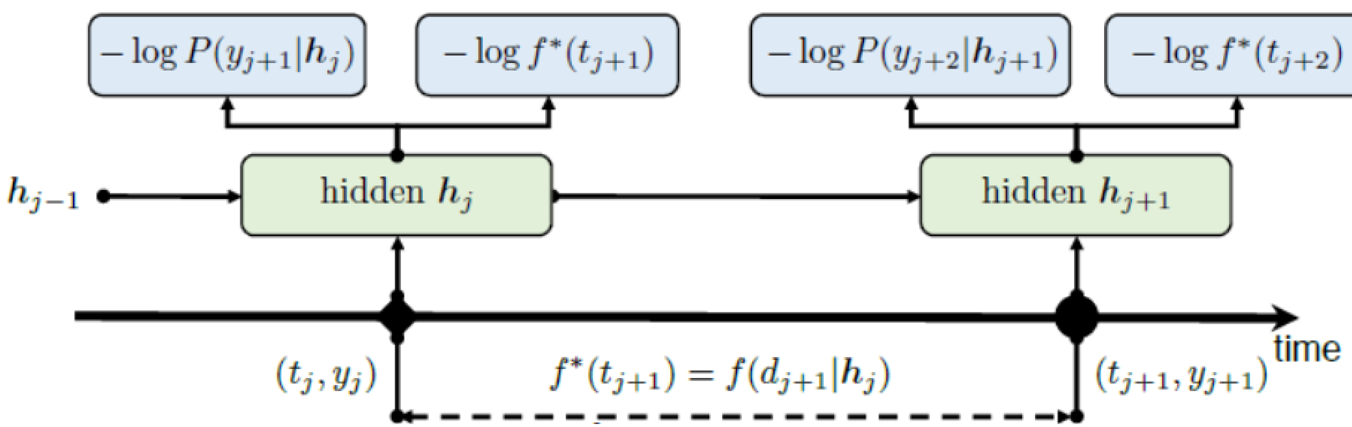
- The second term emphasizes the influence of the current event j
- The last term gives a base intensity level for the occurrence of the next event.
- The exponential function outside acts as a non-linear transformation and guarantees that the intensity is positive.

Deep point process

- MLE for Marked RNN Point Process

$$L = \prod_{j=1}^{n} f(t_{j+1}, y_{j+1}|\boldsymbol{h}_j) = \prod_{j=1}^{n} f(t_{j+1}|\boldsymbol{h}_j) \cdot P(y_{j+1}|\boldsymbol{h}_j)$$

$$\Longrightarrow \log L = \sum_{j=1}^{n} f(t_{j+1}|\boldsymbol{h}_j) + \sum_{j=1}^{n} P(y_{j+1}|\boldsymbol{h}_j)$$



Deep point process

- The density $f^*(t)$ for event $j+1$

$$f^*(t) = f\left(t|h_j\right) = \lambda^*(t)\exp(-\int_{t_j}^{t}\lambda^*(\tau)d\,\tau)$$

$$= \exp\{\mathbf{v}^{\mathbf{T}} \cdot \boldsymbol{h}_j + w\left(t - t_j\right) + b - \frac{1}{w}\left(\exp\left(\mathbf{v}^{\mathbf{T}} \cdot \boldsymbol{h}_j + w\left(t - t_j\right) + b\right) - \exp\left(v^T \cdot h_j + b\right)\right)\}$$

- prediction method1: the exception of t_{j+1}

$$\hat{t}_{j+1} = \int_{t_j}^{\infty} t f^*(t)dt$$

Can not get the analytical result and only numerical integration techniques can be used.

Deep point process

- prediction method2: the simulation way

  Recall the inversed method

For a temporal point process $\{t_1, t_2, \dots, t_n\}$ with conditioned intensity $\lambda^*(t)$ and integrated conditional intensity as:

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

Then $\{\Lambda(t_j)\}$ is a Poisson process with unit intensity. In other words, the integrated conditional intensity between inter-event time

$$\Lambda(t_j, t_{j+1}) = \Lambda(t_{j+1}) - \Lambda(t_j)$$

is exponentially distributed with parameter 1.

Deep point process

- So if given timestamp t_j and function $\Lambda_{t_j}(t_{j+1})$ is known
- Then we can predict t_{j+1} with the inversed method
- Given $s \sim Exp(1)$(i.e. $s = -\log(1-u)$, $u \sim uniform(0,1)$) then

$$\hat{t}_{j+1} = \Lambda_{t_j}^{-1}(s)$$

- Apparently $\Lambda_{t_j}(t_{j+1}) = \Lambda(t_j, t_{j+1})$ satisfies that

$$\Lambda(t_j, t_{j+1}) = \frac{1}{w}(\exp(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + w(t - t_j) + b) - \exp(v^T \cdot h_j + b)) = -\log(1-u)$$

- Then

$$\hat{t}_{j+1} = t_j + \frac{\log\left(\exp\left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right) - w\log(1-u)\right) - \left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right)}{w}$$

Marked RNN Point Process

- Median sampling when $u = 0.5$

$$\hat{t}_{j+1} = t_j + \frac{\log\left(\exp\left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right) + w\log(2)\right) - \left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right)}{w}$$

- Quantile Interval Estimation. Given significance α

$$\hat{t}_{j+1}^{(\frac{\alpha}{2})} = t_j + \frac{\log\left(\exp\left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right) - w\log\left(1 - \frac{\alpha}{2}\right)\right) - \left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right)}{w}$$

$$\hat{t}_{j+1}^{(1-\frac{\alpha}{2})} = t_j + \frac{\log\left(\exp\left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right) - w\log\left(\frac{\alpha}{2}\right)\right) - \left(\mathbf{v}^{\mathrm{T}} \cdot \boldsymbol{h}_j + b\right)}{w}$$

- For example, when given significance $\alpha = 0.1$, it means that there is 90% that the event t_{j+1} is in the interval $[\hat{t}_{j+1}^{(\frac{\alpha}{2})}, \hat{t}_{j+1}^{(1-\frac{\alpha}{2})}]$ in theorem.

Outline：Deep Learning for TPP

3.1 RNN model for TPP

3.2 Adversarial learning for TPP

3.3 Reinforcement learning for TPP

3.4 Embedding for multi-dimensional TPP

Wasserstein-Distance for PP

The Wasserstein distance between distribution of two point processes is:

$$W_1(P_r, P_g) = \inf_{\psi \in \Psi(P_r, P_g)} E_{\{\xi,\rho\}\sim\psi}[\|\xi - \rho\|_*]$$

Ψ denotes the set of all joint distributions whose marginals are P_r, P_g.

The distance between two sequences $\|\xi - \rho\|_*$ need further attention. Take $\xi = \{x_1, \dots, x_n\}$, $\rho = \{y_1, \dots, y_m\}$ and with a permutation σ, w.l.o.g. assuming $n \le m$

$$\|\xi - \rho\|_* = \min_{\sigma} \sum_{i=1}^{n} \|x_i - y_{\sigma(i)}\| + \sum_{i=n+1}^{m} \|s - y_{\sigma(i)}\|$$

s is a fixed limiting point in border

[Xiao et al 2017]

[1] Wasserstein learning of deep generative point process models. In NIPS, 2017.

$\|\xi - \rho\|_*$ is a distance

- It is obvious in nonnegative and symmetric
- triangle inequality : assume $\xi = \{x_1, \dots, x_n\}$, $\rho = \{y_1, \dots, y_k\}$ and $\varsigma = \{z_1, \dots, x_m\}$ where $n \leq k \leq m$, define the permutation

$$\hat{\sigma} := \arg\min_{\sigma} \sum_{i=1}^{n} \|x_i - y_{\sigma(i)}\| + \sum_{i=n+1}^{k} \|s - y_{\sigma(i)}\|$$

- We know that

$$\|\xi - \rho\|_\star = \sum_{i=1}^{n} \|x_i - y_{\hat{\sigma}(i)}\| + \sum_{i=n+1}^{k} \|s - y_{\hat{\sigma}(i)}\|$$

$\|\xi - \rho\|_*$ is a distance

- Therefore, we have that

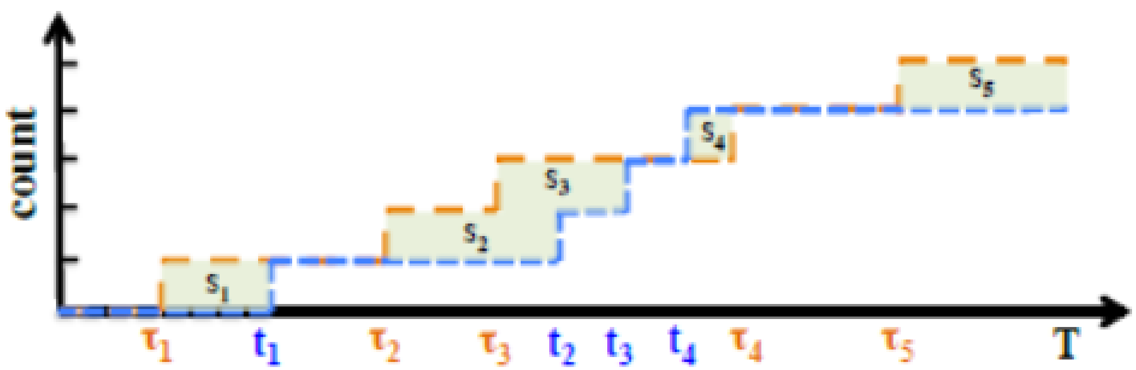
$$
\begin{aligned}
\|\xi - \zeta\|_\star &= \min_\sigma \sum_{i=1}^{n} \|x_i - z_{\sigma(i)}\| + \sum_{i=n+1}^{m} \|s - z_{\sigma(i)}\| \\
&\leq \min_\sigma \sum_{i=1}^{n} \left(\|x_i - y_{\hat{\sigma}(i)}\| + \|y_{\hat{\sigma}(i)} - z_{\sigma(i)}\|\right) + \sum_{i=n+1}^{k} \left(\|s - y_{\hat{\sigma}(i)}\| + \|y_{\hat{\sigma}(t)} - z_{\sigma(i)}\|\right) \\
&\quad + \sum_{i=k+1}^{m} \|s - z_{\sigma(i)}\| \\
&= \|\xi - \rho\|_\star + \min_\sigma \sum_{i=1}^{k} \|y_{\hat{\sigma}(i)} - z_{\sigma(i)}\| + \sum_{i=k+1}^{m} \|s - z_{\sigma(i)}\| \\
&= \|\xi - \rho\|_\star + \min_\sigma \sum_{i=1}^{k} \|y_i - z_{\sigma(\hat{\sigma}^{-1}(i))}\| + \sum_{i=k+1}^{m} \|s - z_{\sigma(i)}\| \\
&= \|\xi - \rho\|_\star + \|\rho - \zeta\|_\star
\end{aligned}
$$

- Same on the real line

Wasserstein-Distance for TPP

Interestingly, in the case of temporal point process in $[0, T]$ for $\xi = \{t_1, \dots, t_n\}$, $\rho = \{\tau_1, \dots, \tau_m\}$ is reduced to

$$\|\xi - \rho\|_* = \sum_{i=1}^{n} |t_i - \tau_i| + (m - n) \times T - \sum_{i=n+1}^{m} \tau_i$$



$\|\cdot\|_*$distance between sequences

Wasserstein-Distance for TPP

Dual Wasserstein distance (event sequence)：

$$\mathrm{W}_1\left(P_r, P_g\right) = \sup_{\|f\|_L \leq 1} E_{\{\xi,\rho\}\sim P_r}[f(\xi)] - E_{\rho\sim P_g}[f(\rho)]$$

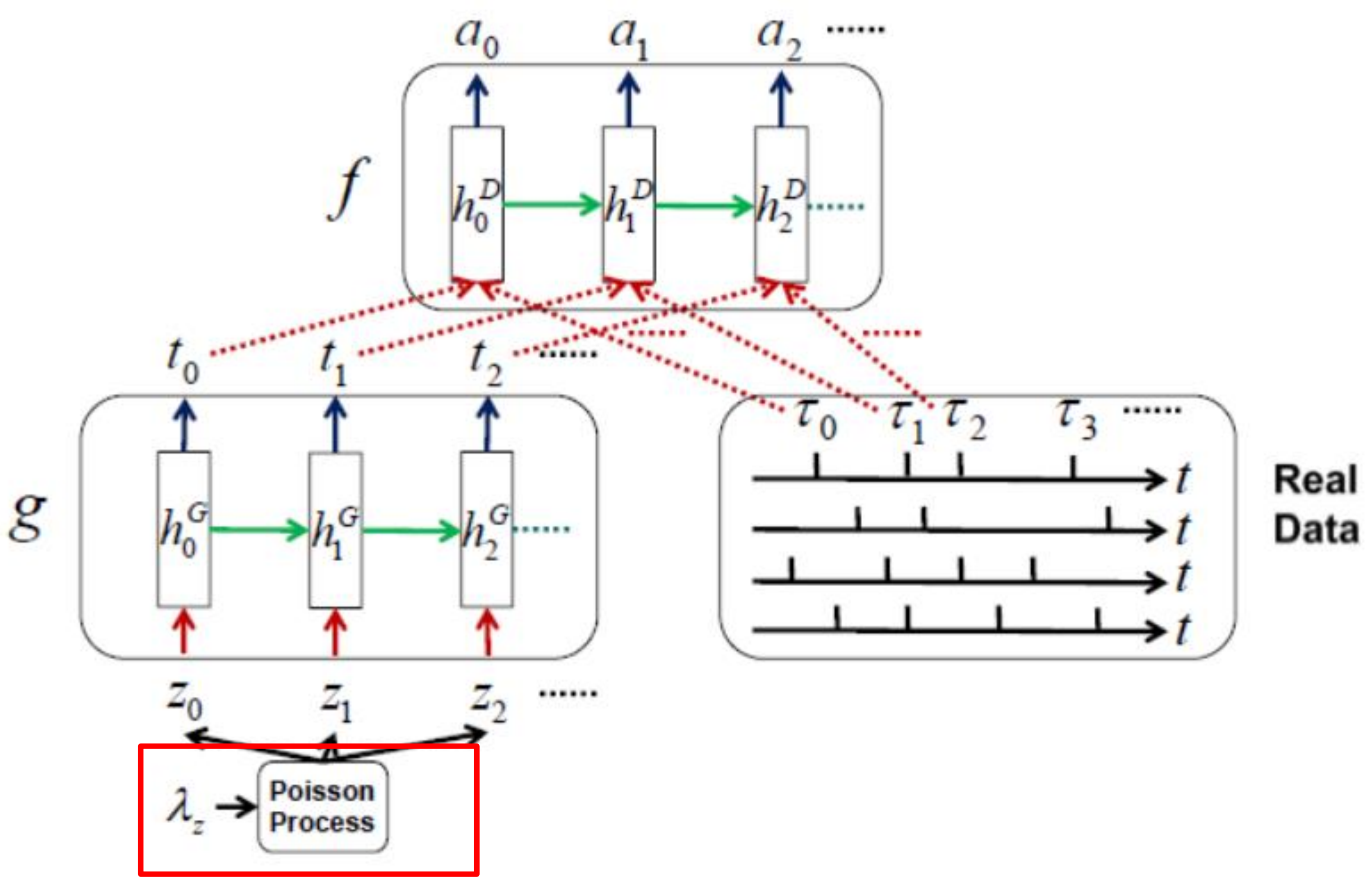
choose a parametric family of functions f_w to approximate f

$$\max_{w\in W, \|f_w\|_L \leq 1} E_{\{\xi,\rho\}\sim P_r}[f_w(\xi)] - E_{\rho\sim P_g}[f_w(\rho)]$$

Combine the objective of the generative model as min $\mathrm{W}_1\left(P_r, P_g\right)$

$$\min_{\theta} \max_{w\in W, \|f_w\|_L \leq 1} E_{\{\xi,\rho\}\sim P_r}[f_w(\xi)] - E_{\varsigma\sim P_z}[f_w(g_\theta(\varsigma))]$$

Wasserstein-Distance for TPP



Wasserstein-GAN for TPP

Final Objective with penalty to guarantee Lipschitz limitation

$$\boxed{\min_{\theta} \max_{w\in\mathcal{W}, \|f_w\|_L\leq 1}} \frac{1}{L}\sum_{l=1}^{L} \boxed{f_w}(\xi_l) - \sum_{l=1}^{L} f_w(\boxed{g_\theta}(\zeta_l)) - \nu \sum_{l,m=1}^{L} \boxed{\left|\frac{|f_w(\xi_l) - f_w(g_\theta(\zeta_m))|}{\boxed{|\xi_l - g_\theta(\zeta_m)|_\star}} - 1\right|}$$

min-max Adverserial

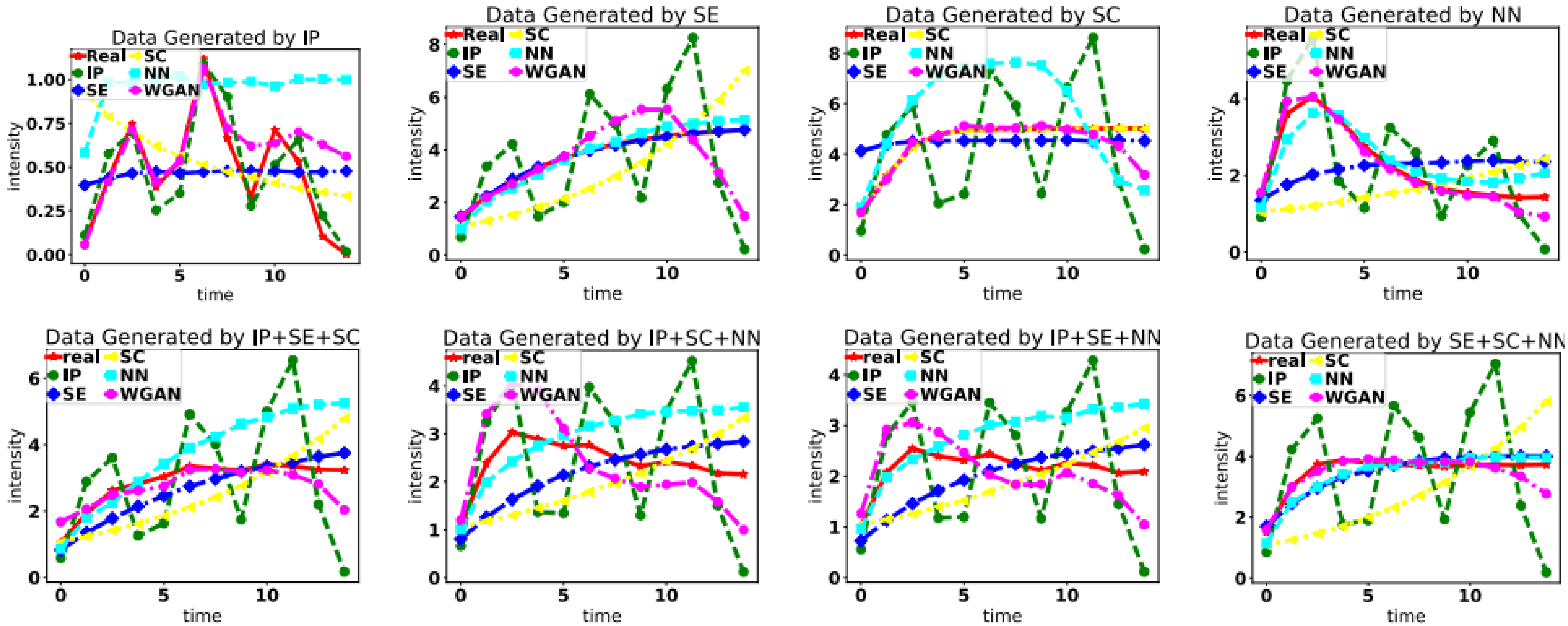
discriminator

generator

Regularization: 1-Lipschitz condition in WGAN

distance between sequences

Synthetic datasets



- **IP**: Inhomogeneous process
- **SE**: Self-exciting process
- **SC**: Self-correcting process
- **NN**: Recurrent Neural Network process

Event sequence learning application case

- **Medical information mining**
  - Data source: MIMIC III
  - https://mimic.physionet.org
- **Job hopping record**
  - Data source: web crawling
  - https://github.com/HongtengXu/Hawkes-Process-Toolkit/blob/master/Data/LinkedinData.mat
- **IPTV（互联网协议电视）log data**
  - TV viewing behavior of 7,100 users
  - https://github.com/HongtengXu/Hawkes-Process-Toolkit/blob/master/Data/IPTVData.mat
- **Stock trading data**
  - 700,000 high-frequency transaction records in one day
  - https://github.com/dunan/NeuralPointProcess/tree/master/data/real/book order.

Data	Estimator						
	MLE-IP	MLE-SE	MLE-SC	MLE-NN	Seq2Seq	SS	CWE
MIMIC	0.25 (2.5e-5)	0.15 (5.3e-4)	0.26 (7.3e-5)	0.19 (2.3e-2)	0.17 (5.3e-3)	0.16 (4.1e-3)	**0.10 (2.5e-3)**
LinkedIn	0.24 (3.1e-4)	0.19 (4.8e-4)	0.17 (9.3e-4)	0.14 (9.1e-3)	0.14 (4.1e-3)	0.12 (8.9e-2)	**0.11 (9.4e-2)**
IPTV	1.46 (3.4e-5)	1.24 (2.8e-5)	1.52 (8.1e-5)	1.21 (2.8e-3)	1.19 (4.2e-2)	1.13 (8.4e-3)	**0.95 (4.9e-3)**
NYSE	2.25 (4.1e-5)	1.96 (6.5e-4)	2.34 (7.3e-5)	1.57 (4.8e-2)	1.55 (2.9e-3)	1.47 (7.3e-3)	**1.23 (2.8e-3)**

Event sequence learning application case

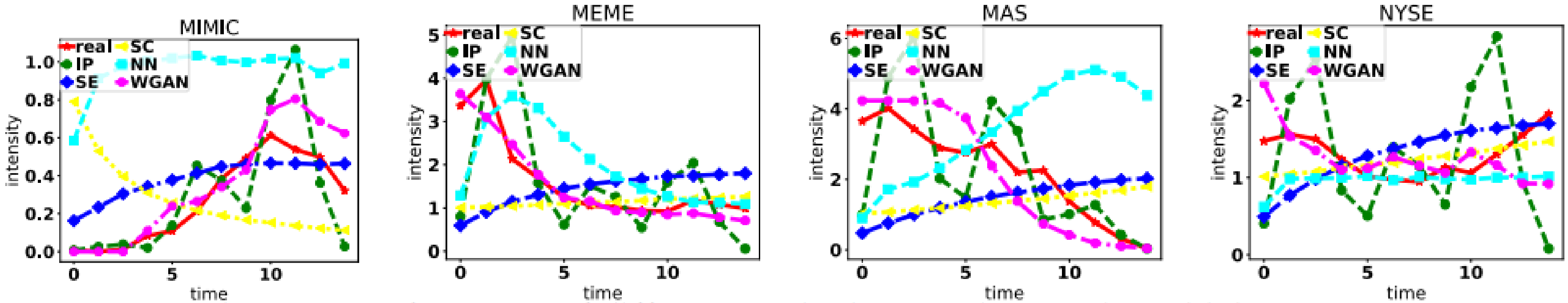


Figure 3: Performance of different methods on various real-world datasets.

Table 2: Deviation of empirical intensity for real-world data.

Data	Estimator				
	MLE-IP	MLE-SE	MLE-SC	MLE-NN	WGAN
MIMIC	0.150	0.160	0.339	0.686	0.122
Meme	0.839	1.008	0.701	0.920	0.351
MAS	1.089	1.693	1.592	2.712	0.849
NYSE	0.799	0.426	0.361	0.347	0.303

Outline：Deep Learning for TPP

3.1 RNN model for TPP

3.2 Adversarial learning for TPP

3.3 Reinforcement learning for TPP

3.4 Graph Embedding for marked TPP

Reinforcement Learning for TPP

- Given a sequence of past events $s_t = \{t_i\}_{t_i<t}$
- The stochastic policy $\pi_\theta(a|s_t)$ samples action a as inter-event time, where $t_{i+1} = t_i + a$
- Relation about intensity

$$\lambda_\theta(t|s_{t_i}) = \frac{\pi_\theta(t-t_i|s_{t_i})}{1-\int_{t_i}^{t}\pi_\theta(\tau-t_i|s_{t_i})d\tau}$$

Here regard policy as the conditional density function f*

- If given a reward function $r(t)$

$$\pi_\theta^* = \arg\max J(\pi_\theta) := E_{\eta\sim\pi_\theta}\left[\sum_{i=1}^{N_T^\eta} r(t_i)\right]$$

[Li et al, 2018]

[1] Learning temporal point processes via reinforcement learning. In NIPS, 2018

Reinforcement Learning for TPP

- However, only the expert's sequences are observed
- Solution: Inverse Reinforcement Learning (IRL)
- Given the expert policy π_E

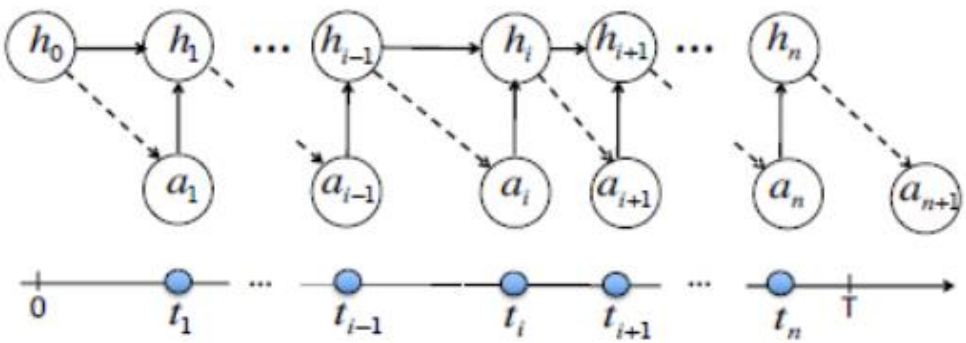
$$r^* = \arg\max \left(E_{\xi\sim\pi_E}\left[\sum_{i=1}^{N_T^{\xi}} r(\tau_i)\right] - \max_{\pi_\theta} E_{\eta\sim\pi_\theta}\left[\sum_{i=1}^{N_T^{\eta}} r(t_i)\right]\right)$$

- The final optimal policy can be obtained by

$$\pi_\theta^* = \mathrm{RL} \circ \mathrm{IRL}(\pi_E)$$

Policy Network

- Adopt the recurrent neural network (RNN) to generate the action $a_i \sim \pi(a|\Theta(h_{i-1})), h_i = \psi(Va_i + Wh_{i-1})$



generator π

- Common distributions such as exponential and Rayleigh distributions would satisfy such constraint

$$\pi(a|\Theta(h)) = \Theta(h) \cdot \exp(-\Theta(h)a) \text{ or } \pi(a|\Theta(h)) = \Theta(h) \cdot \exp(-\Theta(h)a^2/2)$$

Reward Function Class

- The reward function directly quantifies the discrepancy between π_E and π_θ, and it guides the optimal policy
- Choose the reward $r(t)$ in the unit ball in RKHS

$$\phi(\eta) \coloneqq \int_{[0,T)} k(t,\cdot) dN_t^{(\eta)} \qquad \mu_{\pi_\theta} \coloneqq E_{\eta\sim\pi_\theta}[\phi(\eta)]$$

feature mapping from data space to R   mean embedding of the intensity in RKHS

- Using the reproducing property

$$J(\pi_\theta) \coloneqq E_{\eta\sim\pi_\theta}\left[\sum_{i=1}^{N_T^{(\eta)}} r(t_i)\right] = E_{\eta\sim\pi_\theta}\left[\int_{[0,T)} \langle r, k(t,\cdot)\rangle_H dN_t^{(\eta)}\right] = \langle r, \mu_{\pi_\theta}\rangle_H$$

- Also one can obtain $J(\pi_\theta) = \langle r, \mu_{\pi_E}\rangle_H$

Reward Function Class

- Then, reward function can be obtained by

$$\max_{\|r\|_H \le 1} \min_{\pi_\theta} \langle r, \mu_{\pi_E} - \mu_{\pi_\theta} \rangle_H = \min_{\pi_\theta} \max_{\|r\|_H \le 1 x} \langle r, \mu_{\pi_E} - \mu_{\pi_\theta} \rangle_H = \min_{\pi_\theta} \max_{\|r\|_H \le 1} \left\| \mu_{\pi_E} - \mu_{\pi_\theta} \right\|_H$$

where $r^*(\cdot\,|\pi_E, \pi_\theta) = \frac{\mu_{\pi_E} - \mu_{\pi_\theta}}{||\mu_{\pi_E} - \mu_{\pi_\theta}||_H} \propto \mu_{\pi_E} - \mu_{\pi_\theta}$

- **Theorem**: Let the family of reward function be the unit ball in RKHS H, i.e. $||r||_H \le 1$, then the optimal policy obtained by solving

$$\pi_\theta^* = \arg\min_{\pi_\theta} D(\pi_E, \pi_\theta, H)$$

where

$$D(\pi_E, \pi_\theta, H) = \max_{\|r\|_H \le 1} \left(E_{\xi \sim \pi_E} \left[\sum_{i=1}^{N_T^\xi} r(\tau_i) \right] - E_{\eta \sim \pi_\theta} \left[\sum_{i=1}^{N_T^\eta} r(t_i) \right] \right)$$

Reward Function Class

- Finite Sample Estimation. Given L trajectories for expert point processes and M generated by π_θ with embedding μ_{π_E} and μ_{π_θ} estimated by their empirical mean

$$\hat{\mu}_{\pi_E} = \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N_T^{(l)}} k(\tau_i^{(l)},\cdot) \qquad \hat{\mu}_{\pi_\theta} = \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N_T^{(m)}} k(t_i^{(m)},\cdot)$$

- Then for any $t \in [0,T)$, the estimated optimal reward (without normalization) is

$$\hat{r}^*(t) \propto \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N_T^{(l)}} k\left(\tau_i^{(l)},t\right) - \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N_T^{(m)}} k(t_i^{(m)},t)$$

Learning Algorithm

- Learning via Policy Gradient
- Equivalently optimize $D(\pi_E, \pi_\theta, H)^2$ instead of $D(\pi_E, \pi_\theta, H)$

$$\nabla_\theta D(\pi_E, \pi_\theta, H)^2 = E_{\eta\sim\pi_\theta}[\sum_{i=1}^{N_T^{(\eta)}} (\nabla_\theta \log(\pi(a_i|\Theta(h_{i-1})))) \cdot \left(\sum_{i=1}^{N_T^{\eta}} \hat{r}^*(t_i)\right)]$$

- After sampling M trajectories from the current policy, use one trajectory for evaluation and the rest $M-1$ samples to estimate reward function.

Algorithm and framework

Algorithm RLPP: Mini-batch Reinforcement Learning for Learning Point Processes

1. Initialize model parameters θ;
2. For number of training iterations do
   - Sample minibatch of L trajectories of events $\{\xi^{(1)}, \dots, \xi^{(L)}\}$ from expert policy π_E, where $\xi^{(l)} = \{\tau_1^{(l)}, \dots, \tau_{N_T^{(l)}}^{(l)}\}$;
   - Sample minibatch of M trajectories of events $\{\eta^{(1)}, \dots, \eta^{(M)}\}$ from learner policy π_θ, where $\eta^{(m)} = \{t_1^{(m)}, \dots, t_{N_T}^{(m)}\}$;
   - Estimate policy gradient $\nabla_\theta D(\pi_E, \pi_\theta, \mathcal{H})^2$ as

$$\nabla_\theta \frac{1}{M} \sum\nolimits_{m=1}^{M} \left(\sum\nolimits_{i=1}^{N_T^{(m)}} \hat{r}^*(t_i^{(m)}) \log p_\theta(\eta^{(m)}) \right)$$

where $\log p_\theta(\eta^{(m)}) = \sum_{i=1}^{N_T^\eta} (\log \pi_\theta(a_i | \Theta(h_{i-1})))$ is the log-likelihood of the sample $\eta^{(m)}$, and $r^*(t_i^{(m)})$ can be estimated by L expert trajectories and $(M-1)$ roll-out samples without $\eta^{(m)}$

$$\hat{r}^*(t^{(m)}) = \frac{1}{L} \sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{N_T^{(l)}} k(\tau_i^{(l)}, t)$$
$$- \frac{1}{M-1} \sum_{m'=1, m' \neq m}^{M} \sum\nolimits_{j=1}^{N_T^{(m')}} k(t_j^{(m')}, t);$$

   - Update policy parameters as

$$\theta \leftarrow \theta + \alpha \nabla_\theta D(\pi_E, \pi_\theta, \mathcal{H})^2.$$

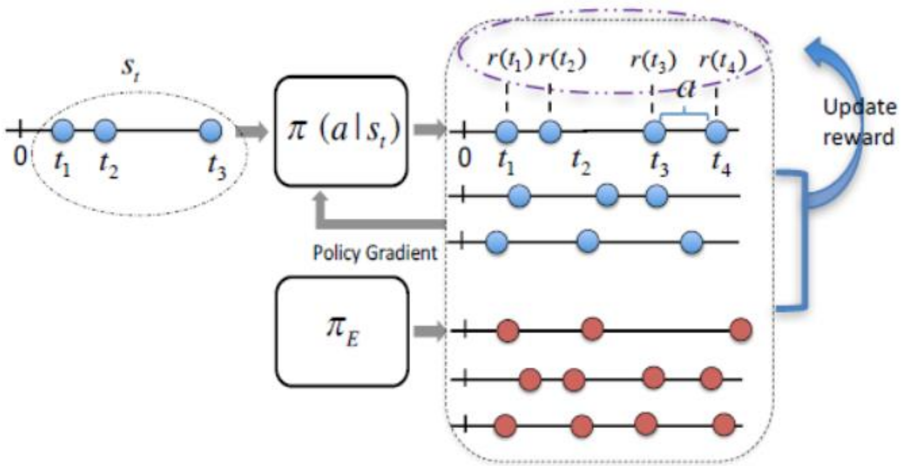
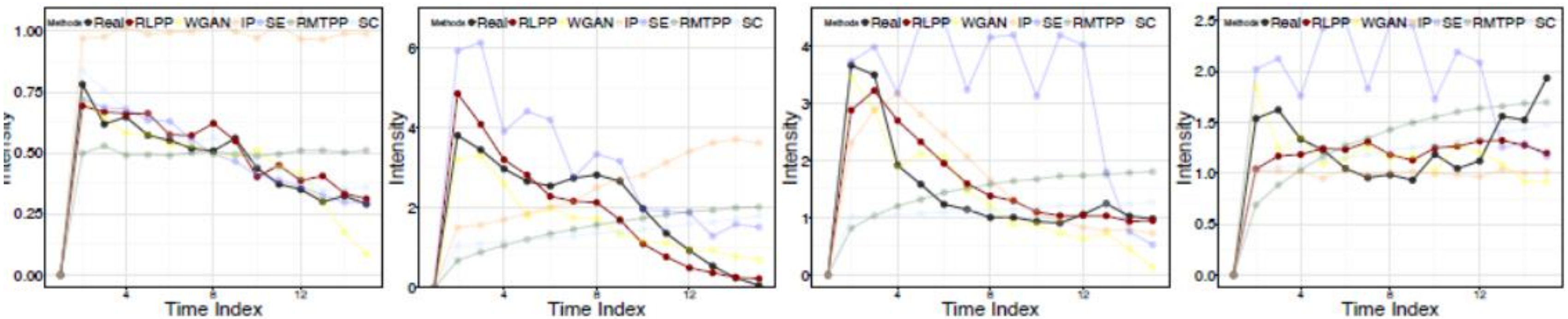
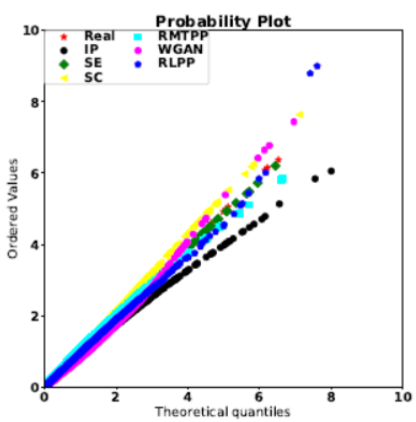


Illustration of the modeling framework.

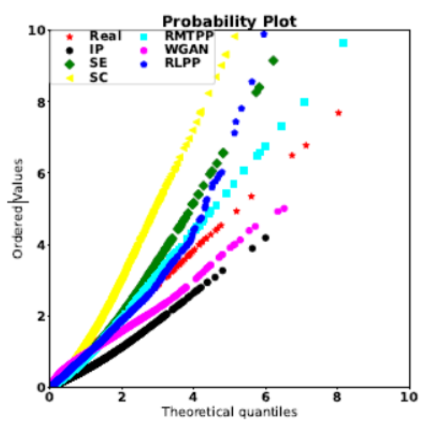
Experiments



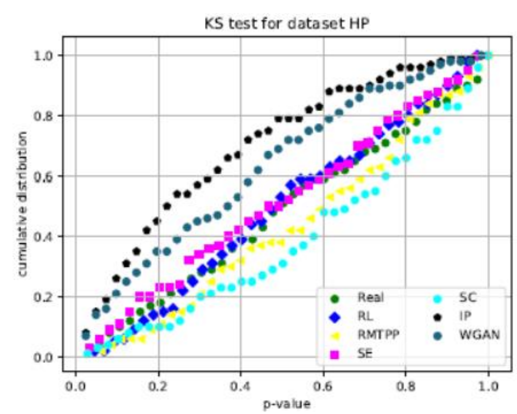
(a) 911 Call (b) MAS (c) MIMIC III (d) NYSE



QQ-plot for HP



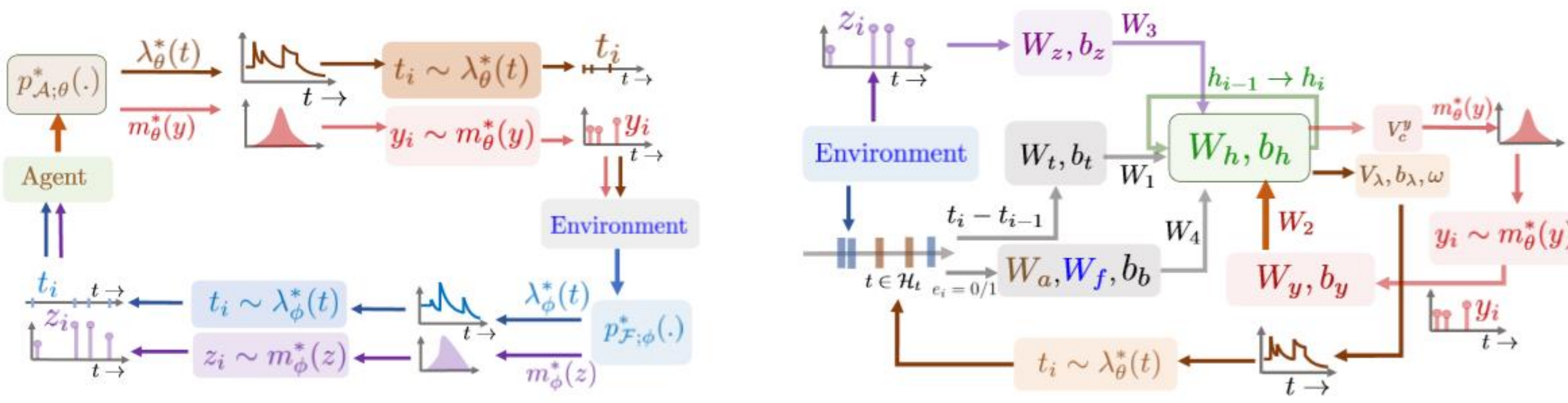
QQ-plot for HP1



KS test results: CDF of p-values.

Other Works for RL in Pont Process

- [Upadhyay et al 2018] use RL to model the marked point process.

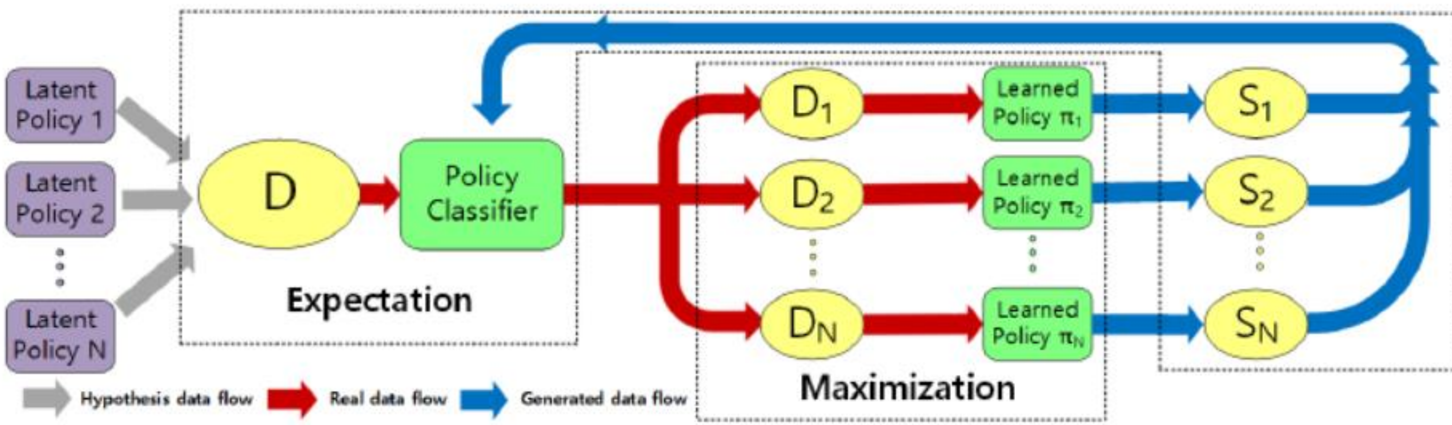


(a) Data and representation (b) Policy parametrization

Deep reinforcement learning of marked temporal point processes. In NIPS, 2018

Other Works for RL in Pont Process

- [Wu et al 2019] cluster the sequences with different temporal patterns into the underlying policies



Reinforcement Learning with Policy Mixture Model for Temporal Point Processes Clustering, https://arxiv.org/abs/1905.12345

Outline： Deep Learning for TPP

3.1 RNN model for TPP

3.2 Adversarial learning for TPP

3.3 Reinforcement learning for TPP

3.4 Graph Embedding for marked TPP

Background

Someone buy something online. For example,

8:30 am, fruits

9:00 am, tickets

11:30 am, lunch

4:00 pm, tickets

4:30 pm, eggs

What is next time?

What is next commodity?

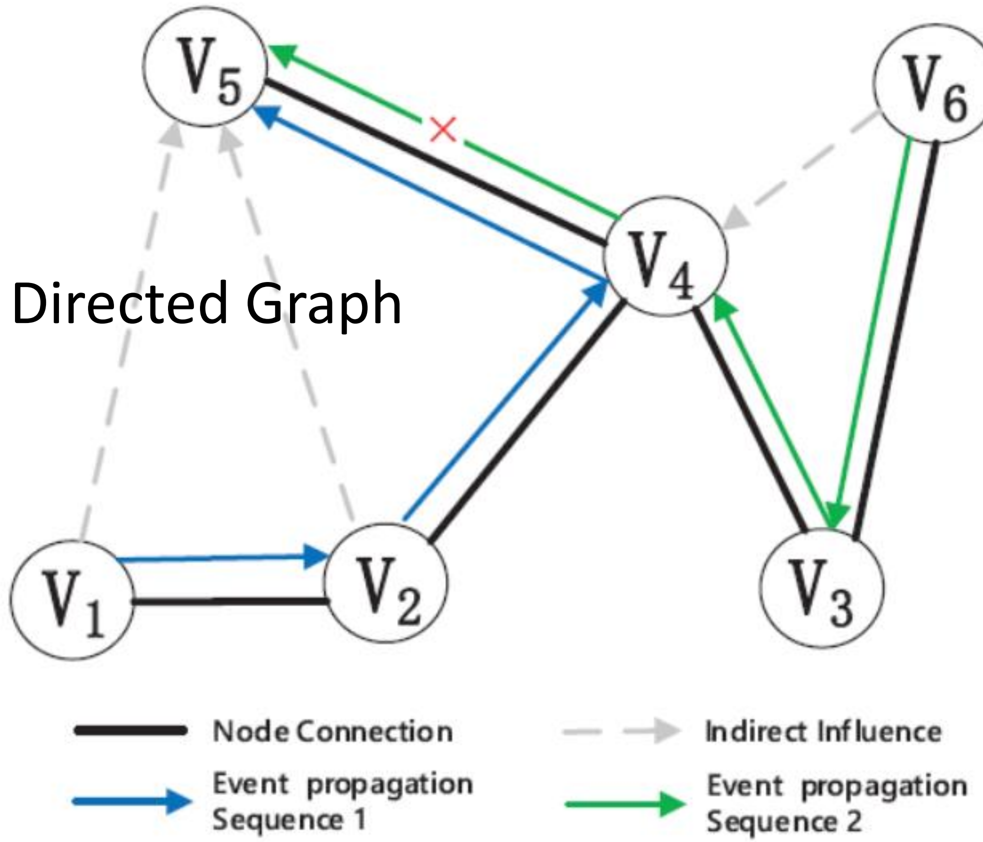
Embedding for multi-dimensional TPP

- Main idea:

  marker → node

  (commodity)

- Graph representation with node embedding for events



[Wu et al 2019]

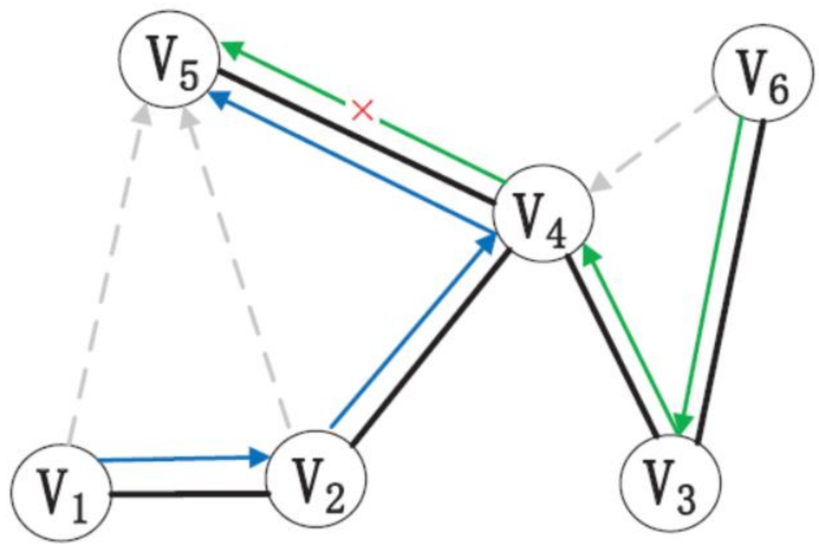
Graph node embedding by MLE (pretrain)

- Graph representation for Directed Graph (pre-trained)
- Capture the edge and node information
- Edge reconstruction probability → MLE

$$p(v_i, v_j) = \frac{1}{1 + \exp(-\boldsymbol{y}_i^{s\mathrm{T}} \boldsymbol{y}_j^e)}$$

- There may be indirect influence form V_1 to V_5
- Get the node representation

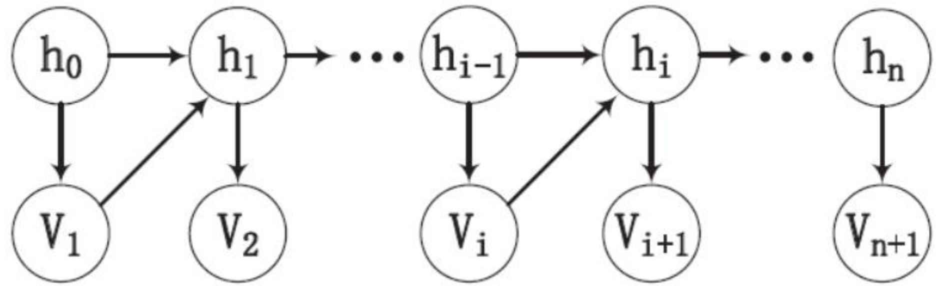
$$\boldsymbol{y}_i = \{\boldsymbol{y}_i^s, \boldsymbol{y}_i^e\}$$



Graph biased TPP

- GBTPP for events (nodes)
- $\boldsymbol{h}_n$ is the hidden units

$$\boldsymbol{h}_{n-1} = \max\{W^v v_{n-1} + W^v y_{n-1}$$



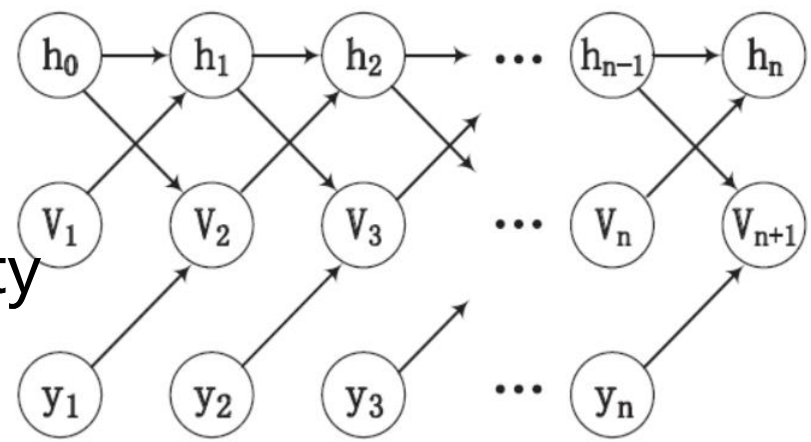
Traditional deep TPP

- Graph bias term

$$b(\boldsymbol{h}_{n-1}, \boldsymbol{y}_n, \boldsymbol{y}_k) = ReLu(\boldsymbol{U}^h_{n,:}\boldsymbol{h}_{n-1})p(\boldsymbol{y}_n, \boldsymbol{y}_k)$$

- Node propagation probability

$$P\left(v_{n+1} = k | \boldsymbol{h}_{n-1}, \boldsymbol{y}_n\right) = \frac{\exp\left(\boldsymbol{V}^h_{k,:}\boldsymbol{h}_{n-1} + b(\boldsymbol{h}_{n-1}, \boldsymbol{y}_n, \boldsymbol{y}_k) + b^h_k\right)}{\sum_{k=1}^{V}\exp\left(\boldsymbol{V}^h_{k,:}\boldsymbol{h}_{n-1} + b(\boldsymbol{h}_{n-1}, \boldsymbol{y}_n, \boldsymbol{y}_k) + b^h_k\right)}$$



GBTPP based on $\{\boldsymbol{y}_i\}$

Graph biased TPP

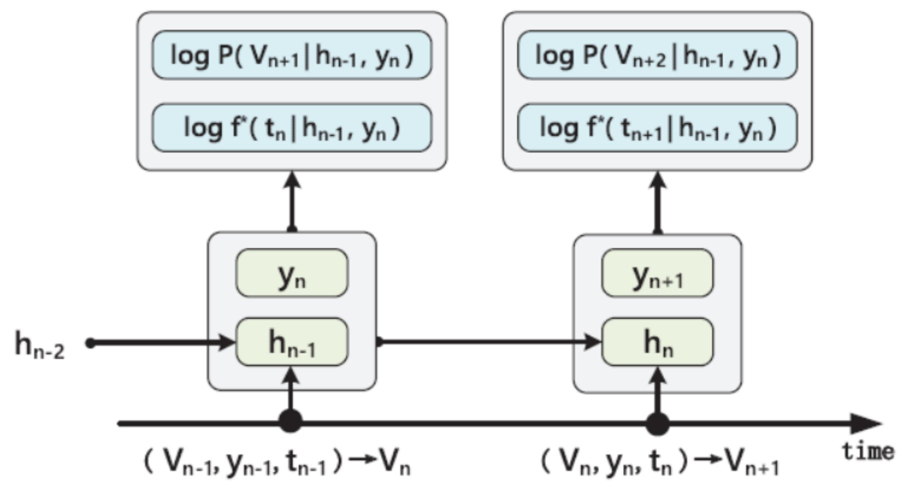
- For conventional intensity function

$$\lambda^*(t) = \exp(\underbrace{\mathbf{v}^{h\mathrm{T}} \cdot \boldsymbol{h}_{n-1}}_{\text{past history influence}} + \underbrace{\mathbf{v}^{y\mathrm{T}} \cdot \boldsymbol{y}_n}_{\text{direct node influence}} + \underbrace{w^t\left(t - t_j\right)}_{\text{Exciting kernel function}} + \underbrace{b}_{\text{base intensity}})$$

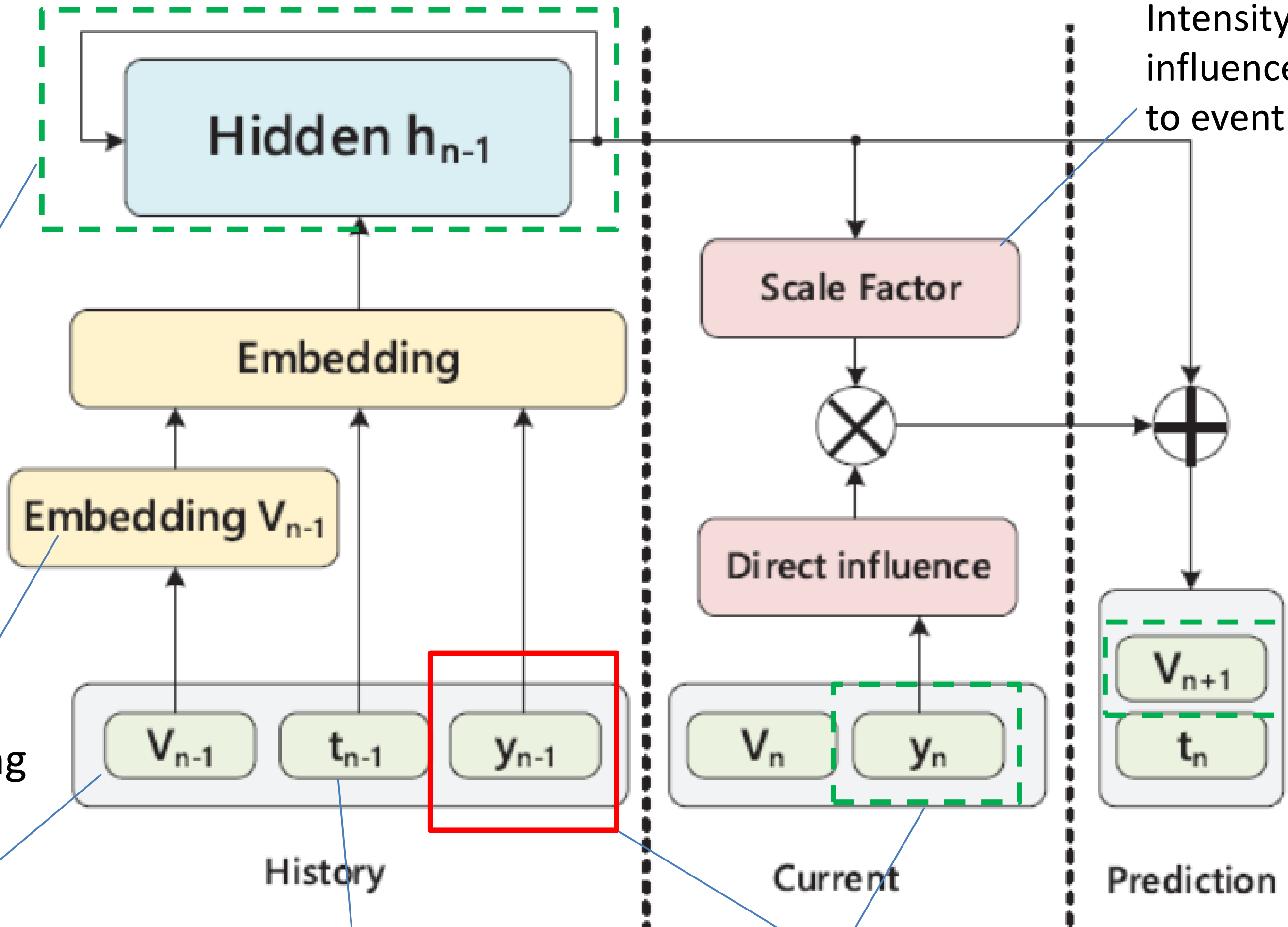
- Density function

$$f^*(t) = \lambda^*(t)\exp(-\int_{t_j}^{t} \lambda^*(\tau)d\,\tau)$$

$$= \exp\{\mathbf{v}^{h\mathrm{T}} \cdot \boldsymbol{h}_{n-1} + \mathbf{v}^{y\mathrm{T}} \cdot \boldsymbol{y}_n + w\big(t$$



Graph biased TPP



Intensity of direct influence is related to event history

Event history embedding

Marker's embedding

Marker's one-hot representation

Time feature, e.g., $t_{n-1} = T_n - T_{n-1}$

Node Embedding by pretrain (constant)

Experiments

- Node Prediction Accuracy & Time Prediction RMSE

✓ MC: Markov Chain (1st 2nd 3rd -order)

✓ PP: homogeneous Poisson Process

✓ HP: Hawkes Process

✓ SCP: Self-Correcting Process

✓ CTMC: Continuous-Time Markov Chain

✓ RMTPP: Recurrent Marked Temporal Point Process

✓ **GBTPP: Graph Biased Temporal Point Process)**

Model	MC-1	MC-2	MC-3	CTMC	RMTPP	GBTPP
Synthetic	17.46 (2.24)	25.27 (2.53)	33.74 (1.87)	32.08 (2.74)	46.82 (1.38)	**47.26** (1.55)
Higgs	10.92 (2.06)	14.60 (1.44)	16.35 (1.73)	17.41 (2.58)	22.26 (1.80)	**24.59** (1.29)
Meme	15.72 (2.21)	20.05 (2.14)	22.93 (1.59)	25.56 (2.17)	32.14 (1.52)	**35.82** (1.73)

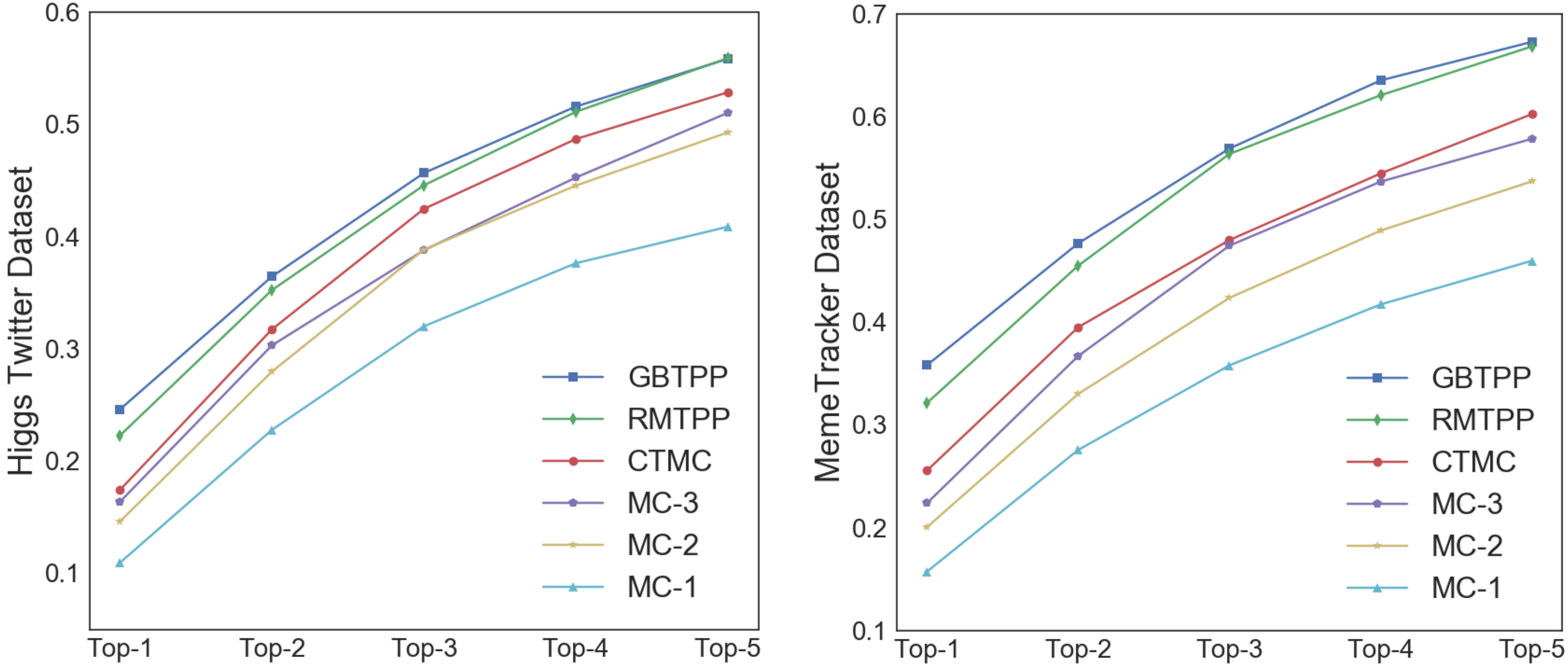
Node prediction accuracy and standard deviation

Model	PP	HP	SCP	CTMC	RMTPP	GBTPP
Synthetic	3.457 (0.374)	2.164 (0.283)	2.845 (0.317)	3.420 (0.265)	1.852 (0.241)	**1.728** (0.228)
Higgs	3.267 (0.381)	2.518 (0.346)	2.343 (0.369)	2.355 (0.335)	1.741 (0.272)	**1.396** (0.264)
Meme	2.361 (0.412)	1.958 (0.368)	1.484 (0.276)	1.762 (0.347)	1.059 (0.254)	**0.825** (0.227)

Time prediction RMSE and standard deviation

Experiments

- Top-5 Node Prediction Accuracy



Higgs Twitter Dataset

MemeTracker Dataset

Reference

[1] L. Li and H. Zha. Learning parametric models for social infectivity in multidimensional Hawkes processes. In AAAI, 2014

[2] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In KDD, 2016.

[3] K. Zhou, L. Song and H. Zha. Learning Social Infectivity in Sparse Low-rank Networks Using Multi-dimensional Hawkes Processes. In AISTATS 2013

[4] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha. Wasserstein learning of deep generative point process models. In NIPS, 2017.

[5] S. Li, S. Xiao, S. Zhu, N. Du, Y. Xie, and L. Song. Learning temporal point processes via reinforcement learning. In NIPS, 2018.

[6] W. Wu, H. Zha. Modeling Event Propagation via Graph Baised Point Process. Submitted to TNNLS 2019

[7] U. Upadhyay, A. De, and M. G. Rodriguez. Deep reinforcement learning of marked temporal point processes. In NIPS, 2018.

[8] W. Wu, J. Yan, X. Yang, H. Zha. Reinforcement Learning with Policy Mixture Model for Temporal Point Processes Clustering, https://arxiv.org/abs/1905.12345

Thanks