Fast and Flexible Temporal Point Processes with Triangular Maps

Oleksandr Shchur, Nicholas Gao,
Marin Biloš, Stephan Günnemann

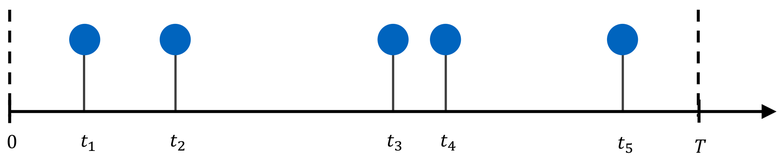
NEURAL INFORMATION PROCESSING SYSTEMS

tl;dr

- Represent temporal point process densities as transformations
- Define flexible, tractable and efficient (parallelizable) transformations on sequences as compositions of simple maps
- Reparametrization trick + differentiable relaxation enable learning TPPs with sampling-based losses
- Use TPPs for variational inference in Markov jump processes

What is a temporal point process?

Temporal point process (TPP) is a probability distribution over variable-length event sequences $\boldsymbol{t} = (t_1, \dots, t_N)$ on an interval $[0, T]$

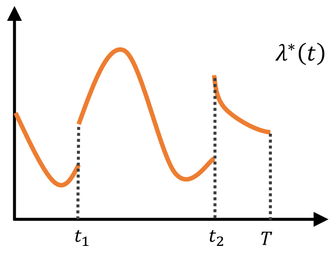


How are TPPs usually defined?

Conditional intensity function $\lambda^*(t)$ describes the rate of arrival new events at time t given the history

$$p(\boldsymbol{t}) = \left(\prod_i^N \lambda^*(t_i)\right) \exp\left(-\int_0^T \lambda^*(u)du\right)$$

State-of-the-art approaches parametrize $\lambda^*(t)$ with autoregressive neural nets



✔ Autoregressive models are flexible
✘ Sequential sampling is extremely slow

Research question

How can we define TPPs that are

- as flexible as autoregressive models?
- where we can both sample & compute likelihood in parallel?

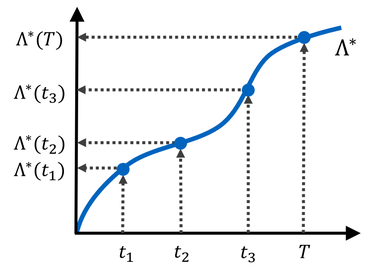
Defining TPPs as transformations

Integrated intensity function (compensator)

$$\Lambda^*(t_i) \coloneqq \Lambda(t_i|t_1, \dots, t_{i-1}) = \int_0^t \lambda^*(u)du$$

Random time change theorem

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} = \begin{bmatrix} \Lambda(t_1) \\ \Lambda(t_2|t_1) \\ \vdots \\ \Lambda(t_N|t_1, \dots, t_{N-1}) \end{bmatrix} = \boldsymbol{F}\left(\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}\right)$$



The sequence $\boldsymbol{z} = \boldsymbol{F}(\boldsymbol{t})$ is distributed according to the Standard Poisson Process

The triangular map $\boldsymbol{F}$ defines the TPP density

$$p(\boldsymbol{t}) = \underbrace{\left(\prod_{i=1}^N \frac{\partial \Lambda^*(t_i)}{\partial t_i}\right)}_{\text{Jacobian determinant of } \boldsymbol{F}} \underbrace{\exp\left(-\Lambda^*(T)\right)}_{\text{Standard Poisson process density at } \boldsymbol{z} = \boldsymbol{F}(\boldsymbol{t})}$$

Inverse transform sampling for TPPs

1. Sample $\boldsymbol{z}$ from the standard Poisson process
2. Transform $\boldsymbol{t} = \boldsymbol{F}^{-1}(\boldsymbol{z})$
3. Discard events t_i after T

Main idea

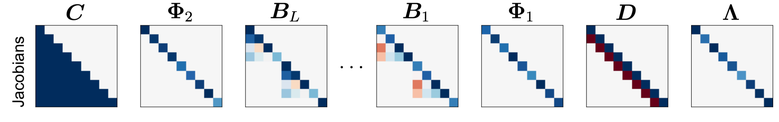
Define TPPs by directly specifying the triangular map $\boldsymbol{F}$

How should we parametrize $\boldsymbol{F}$?

The parametrization of $\boldsymbol{F}$ must be

- flexible – able to represent different distributions
- fast – we can compute $\boldsymbol{F}(\boldsymbol{t})$ and $\boldsymbol{F}^{-1}(\boldsymbol{z})$ analytically in $O(N)$ parallel operations

Solution: Define $\boldsymbol{F}$ as a composition of easy-to-invert transformations



→ New efficient parametrizations for existing TPP models
→ TriTPP: A new flexible model where all operations can be done in parallel

Learning TPPs with sampling-based losses

Sampling losses are common in variational inference, reinforcement learning

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{t}\sim p_{\boldsymbol{\theta}}(\boldsymbol{t})}[g(\boldsymbol{t})]$$

We show how to optimize such objectives thanks to

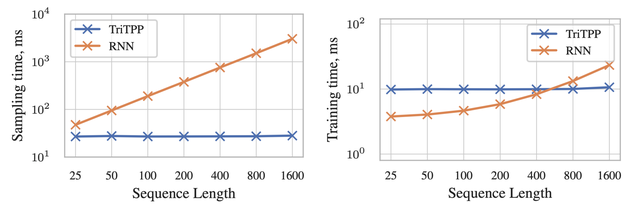
1. Reparametrization trick for TPPs that allows computing $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ with Monte Carlo

$$\mathbb{E}_{\boldsymbol{t}\sim p_{\boldsymbol{\theta}}(\boldsymbol{t})}[g(\boldsymbol{t})] = \mathbb{E}_{\boldsymbol{z}\sim \text{SPP}}\left[g\left(\boldsymbol{F}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{z})\right)\right]$$

2. Differentiable relaxation to $\mathcal{L}$ that removes discontinuities w.r.t. $\boldsymbol{\theta}$
   → Instead of discarding $t_i > T$, use indicator functions
   → Sigmoid relaxes the indicator function

$$\mathbb{1}(t_i < T) \approx \sigma(T - t_i)$$

Scalability analysis



Density estimation

Test set negative log-likelihood

	Hawkes1	Hawkes2	SC	IPP	MRP	RP	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
IPP	1.06	1.03	1.00	**0.71**	0.70	0.89	-0.06	-1.59	-4.08	**-0.68**	1.60	0.62	-0.05
RP	0.65	0.08	0.94	0.85	0.68	**0.24**	0.12	-2.08	-4.00	-0.58	1.20	0.67	-0.02
MRP	0.65	0.07	0.93	**0.71**	0.36	0.25	-0.83	-2.13	-4.38	**-0.68**	1.23	**0.61**	**-0.10**
Hawkes	**0.51**	0.06	1.00	0.86	0.98	0.39	0.11	**-2.40**	-4.19	-0.64	**1.04**	0.69	0.01
RNN	0.52	**-0.03**	**0.79**	0.73	0.37	**0.24**	-1.96	**-2.40**	**-4.89**	-0.66	1.08	0.67	-0.08
TriTPP	0.56	0.00	0.83	**0.71**	**0.35**	**0.24**	**-2.41**	-2.36	-4.49	-0.67	1.06	0.64	-0.09

Maximum mean discrepancy between model samples and the test set

	Hawkes1	Hawkes2	SC	IPP	MRP	RP	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
IPP	0.08	0.09	0.58	**0.02**	0.15	0.07	**0.01**	0.10	0.21	0.10	0.16	0.15	0.16
RP	0.06	0.06	1.13	0.34	1.24	**0.01**	0.46	0.07	0.18	0.57	0.14	0.16	0.23
MRP	0.05	0.06	0.50	**0.02**	0.11	0.02	0.12	0.09	0.20	0.09	0.13	0.13	0.16
Hawkes	0.02	0.04	0.58	0.36	0.65	0.05	0.16	**0.04**	0.35	0.20	0.20	0.20	0.32
RNN	**0.01**	**0.02**	**0.19**	0.09	0.17	**0.01**	0.23	**0.04**	**0.09**	0.13	**0.08**	0.19	0.18
TriTPP	0.03	0.03	0.23	**0.02**	**0.08**	**0.01**	0.16	0.07	0.16	**0.08**	**0.08**	**0.12**	**0.14**

Variational inference in Markov jump processes

- Goal: Infer the posterior distribution over the latent state in a continuous-time discrete-state system
- Approximate posterior over the jump times modeled with TriTPP
- ELBO can be optimized thanks to fast sampling + reparametrization trick + differentiable relaxation