Modeling Continuous-time Event Data with Temporal Point Processes

Oleksandr Shchur

Technical University of Munich

Berlin Time Series Analysis Meetup – October 12th, 2021

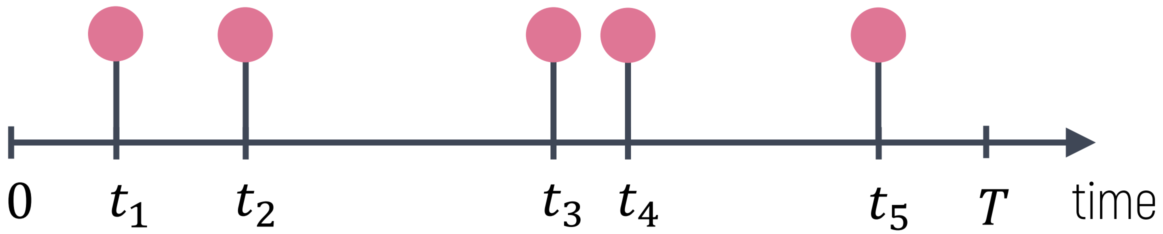
Agenda

- Basics
- Applications
- Old-school TPP models
- Neural TPP models
- Training TPPs

Temporal point process (TPP)

Probability distribution over variable-length continuous-time event sequences

- Hospital visits
- Financial transactions
- Social media posts

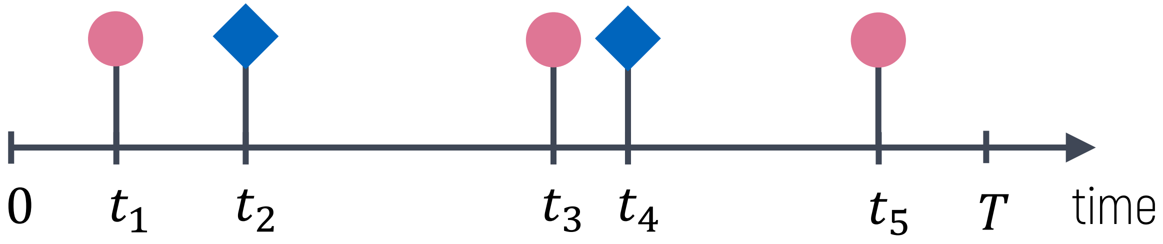


Marked TPPs

Marks – additional features associated with each event, such as

- User ID in the social network
- Magnitude of an earthquake
- Location of a disease outbreak

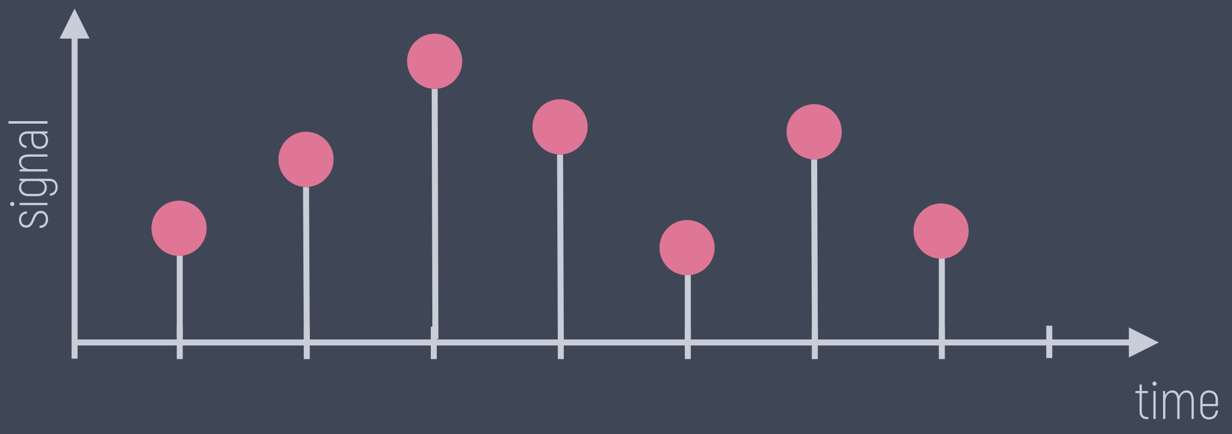
Marks can be continuous, discrete, vector-valued, …



Time series vs. TPPs

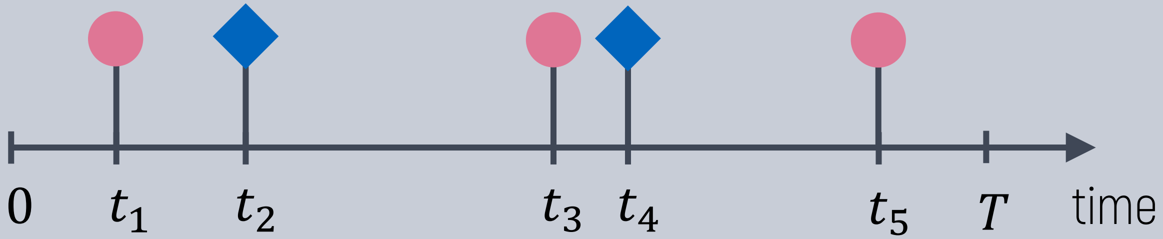
Time series

- Signal measured at regular intervals



Temporal point process

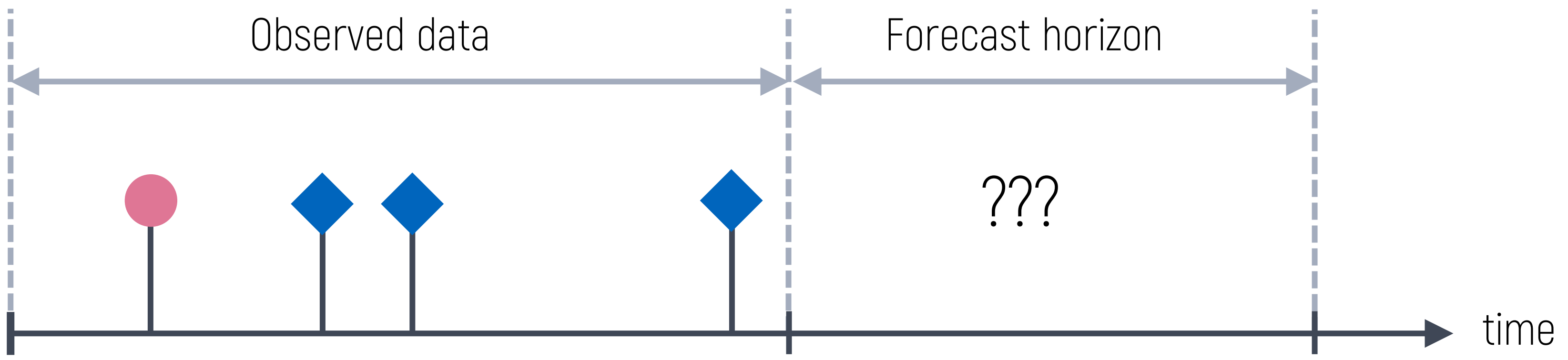
- We model the arrival times
- The number of events is random



Applications

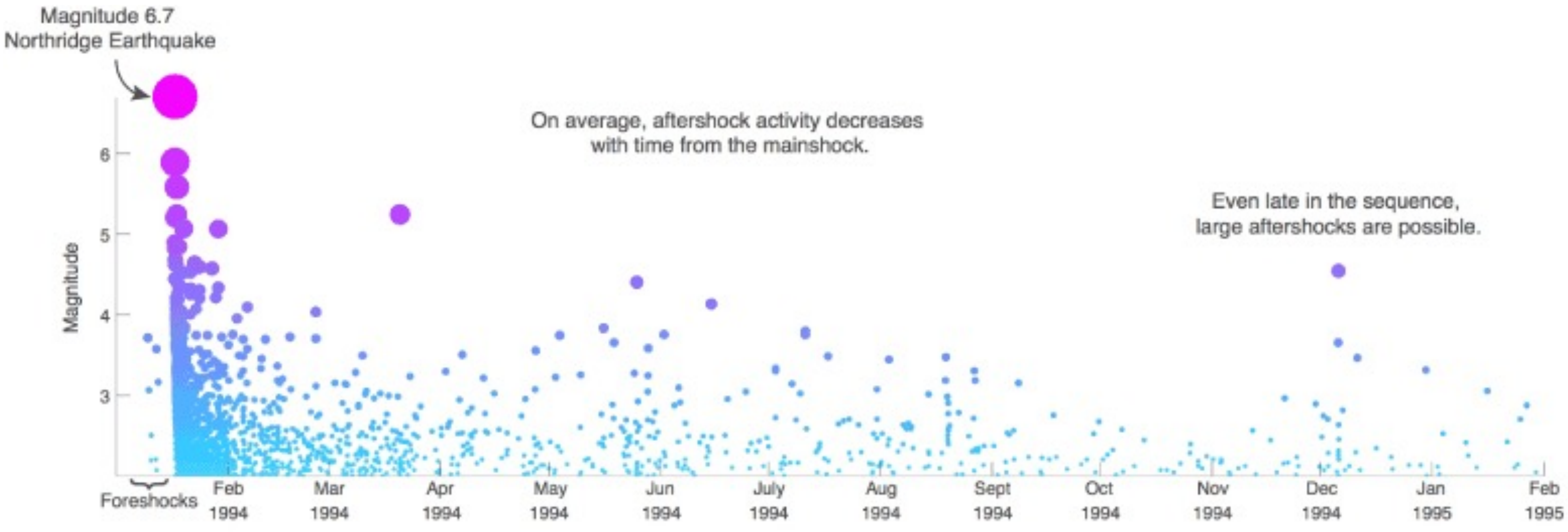
Application: Prediction

- What will be the type of the next event?
- When will the next event of type ● happen?
- How many events of type ◆ will happen in the future?



Application: Earthquake forecasting

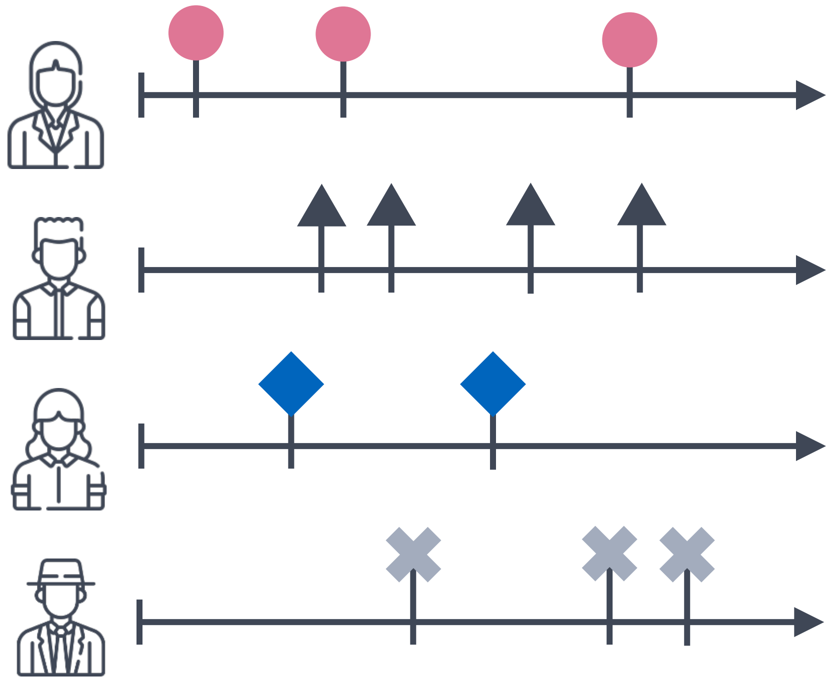
- How many aftershocks of magnitude ≥4 do we expect in the next month?



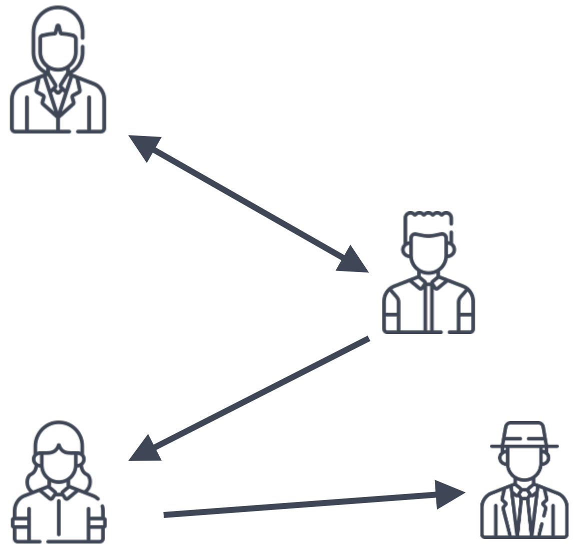
[https://earthquake.usgs.gov/data/oaf/overview.php]

Application: Structure discovery

Observed event sequences



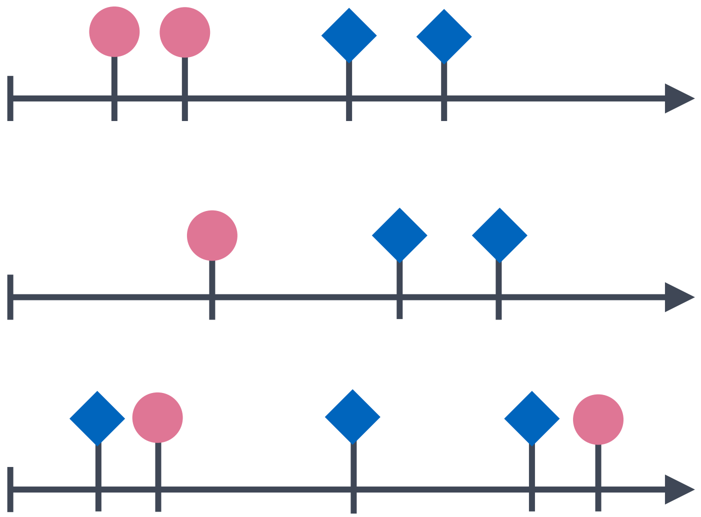
Influence structure



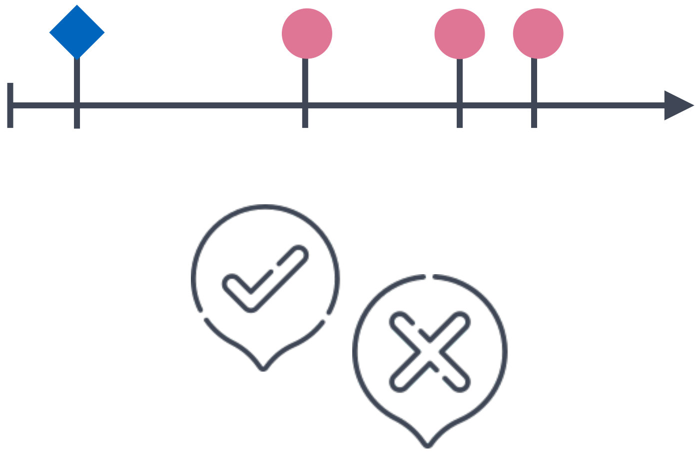
[Linderman, Adams, ICML 2014; Xu, Farajtabar, Zha, ICML 2016]

Application: Anomaly detection

Normal data



Is a new sequence
normal or anomalous?



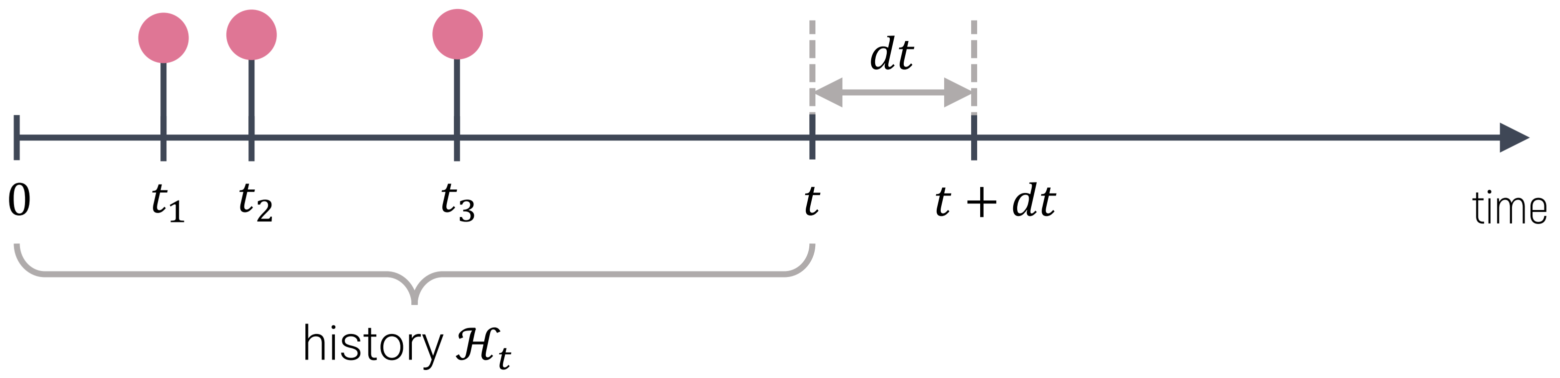
[Shchur, Türkmen, Januschowski, Gasthaus, Günnemann, NeurIPS 2021]

Old-school TPPs

How can we describe a TPP?

- A TPP is fully specified by its conditional intensity function

$$\lambda(t|\mathcal{H}_t) = \lim_{dt \to 0} \frac{\Pr(\text{next event} \in [t, t + dt)|\mathcal{H}_t)}{dt}$$

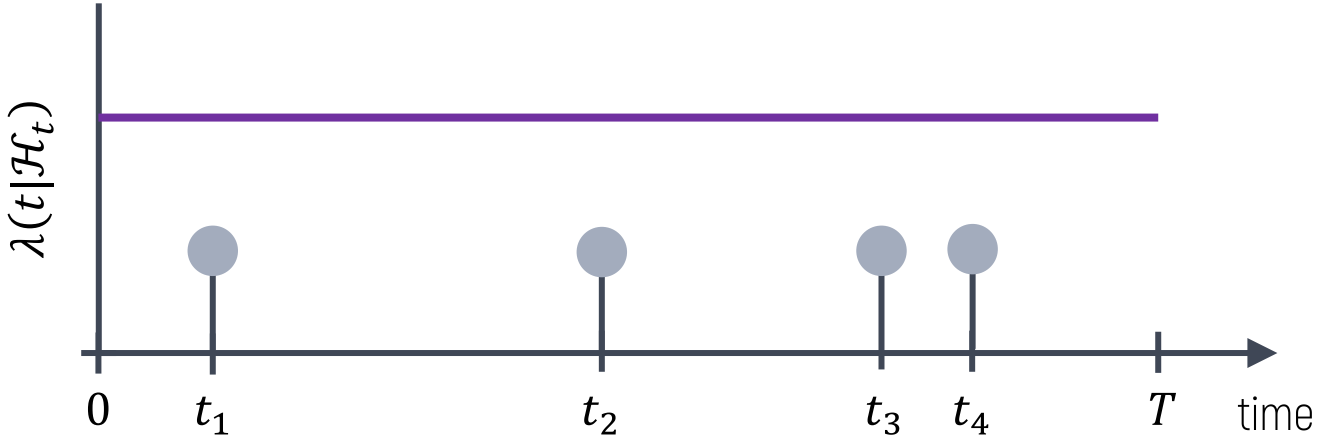


Homogeneous Poisson process

- Simplest possible model – <u>constant intensity</u>

$$\lambda(t|\mathcal{H}_t) = \mu$$

- Events are independent
- Rate of arrival is constant

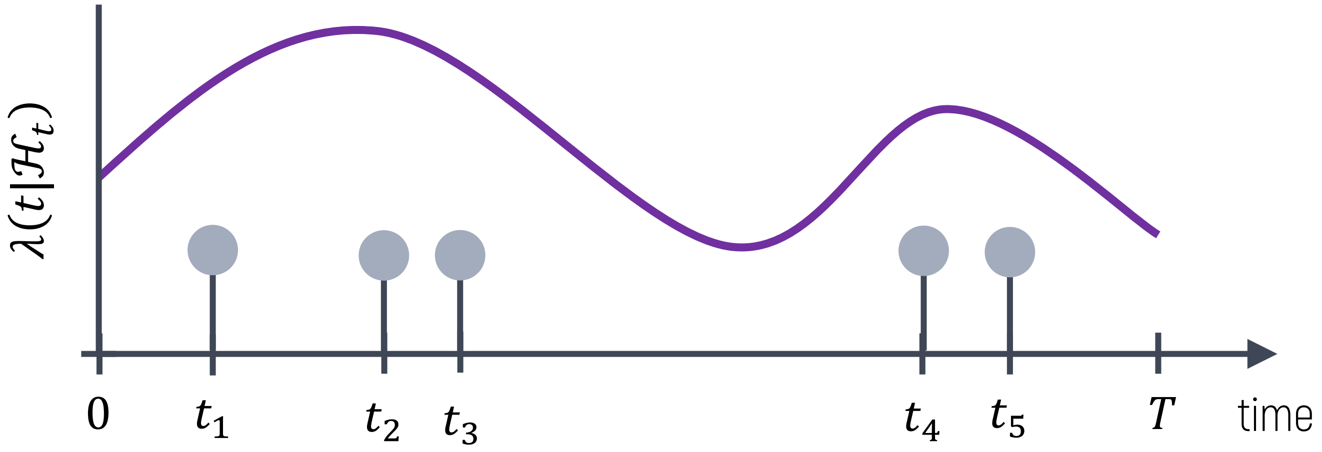


Inhomogeneous Poisson process

- Intensity changes over time but is independent of history

$$\lambda(t|\mathcal{H}_t) = g(t)$$

- Captures global trends

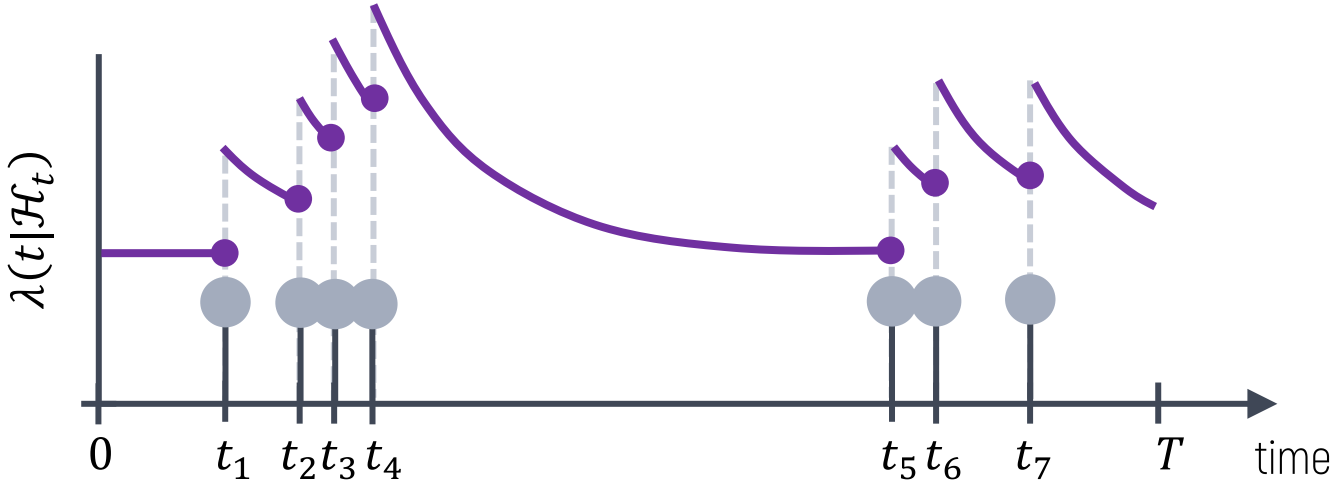


Hawkes process

- Intensity increases after each event, then decays to the baseline

$$\lambda(t|\mathcal{H}_t) = \mu + \sum_{t_j \in \mathcal{H}_t} \alpha \exp\left(-\beta(t - t_j)\right)$$

- Events are clustered ("bursty")

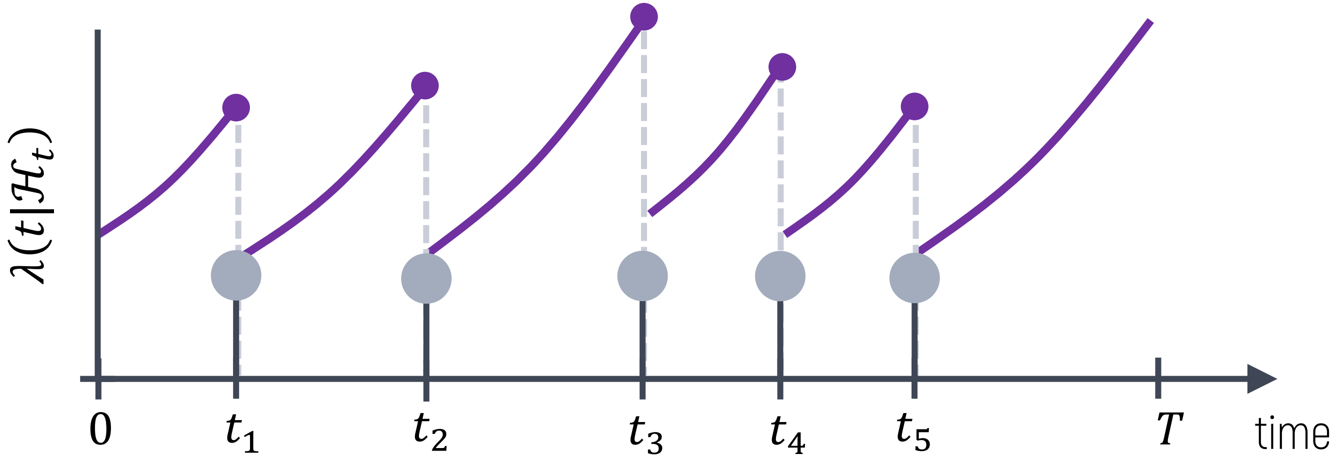


Self-correcting process

- Intensity accumulates over time, drops after each event

$$\lambda(t|\mathcal{H}_t) = \exp\left(\mu t - \sum_{t_j \in \mathcal{H}_t} \alpha\right)$$

- Events are evenly-spaced



Overview of conventional TPPs

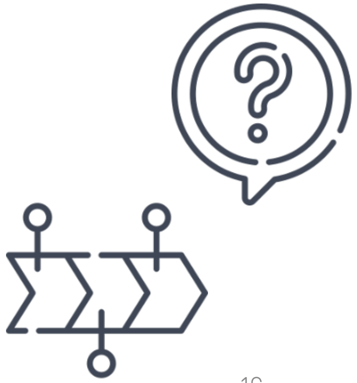
- Conditional intensity $\lambda(t|\mathcal{H}_t)$ fully defines the TPP

- Simple parametric intensity functions

✔ Interpretable

✘ Limited flexibility

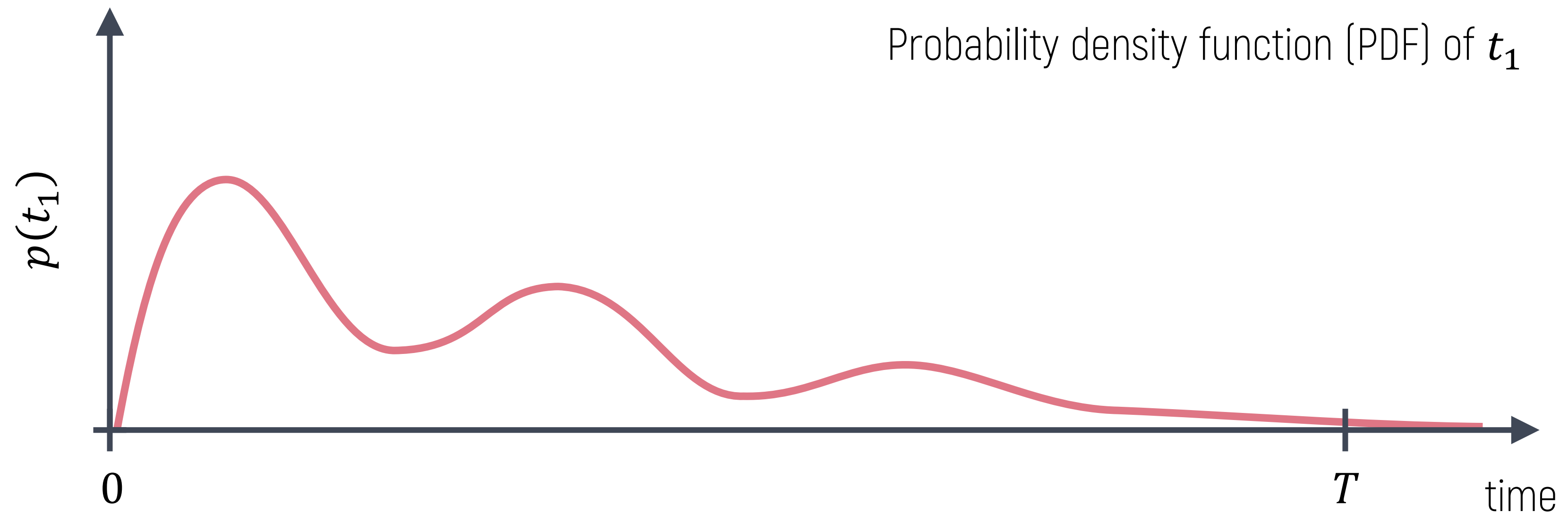
- How do we define <u>flexible</u> TPPs that capture complex dependencies?



Neural TPPs:
Autoregressive models

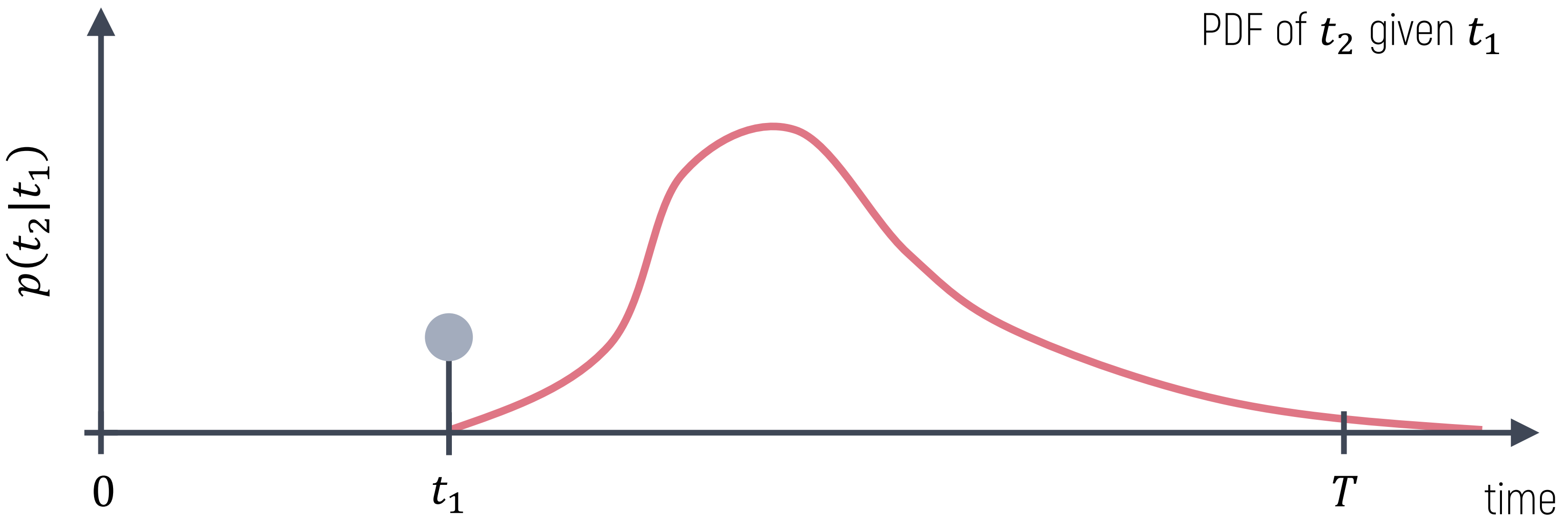
TPP as an autoregressive model

- We can equivalently define a TPP by modeling conditional distributions



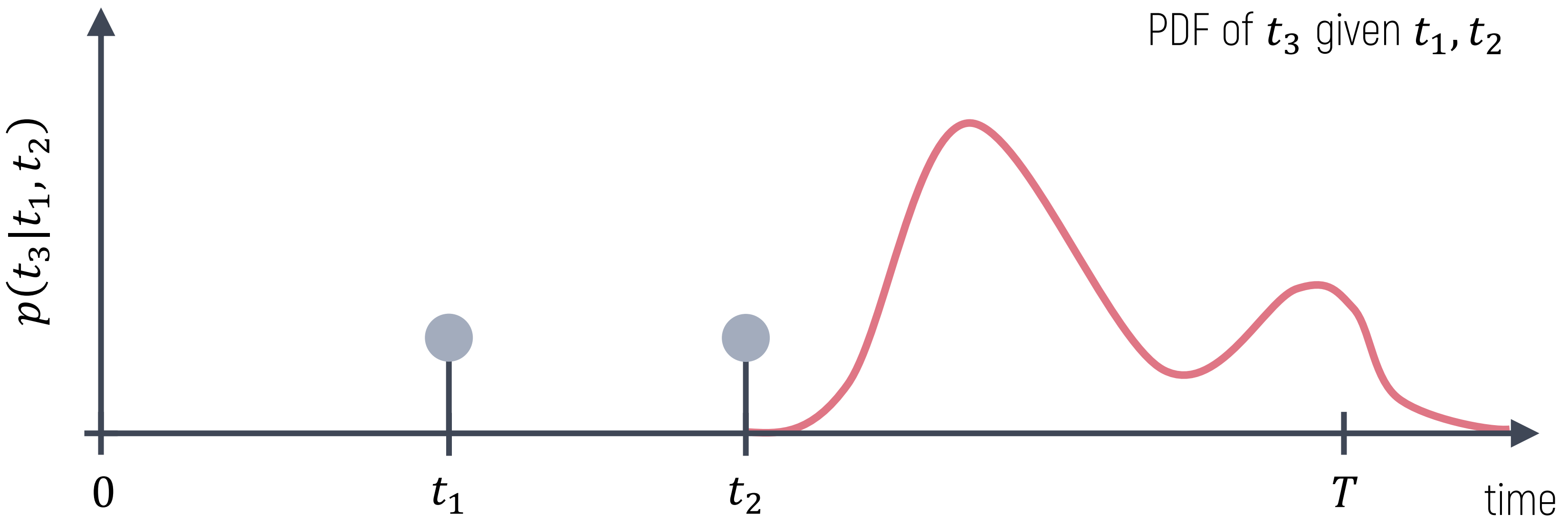
TPP as an autoregressive model

- We can equivalently define a TPP by modeling <u>conditional distributions</u>



TPP as an autoregressive model

- We can equivalently define a TPP by modeling conditional distributions



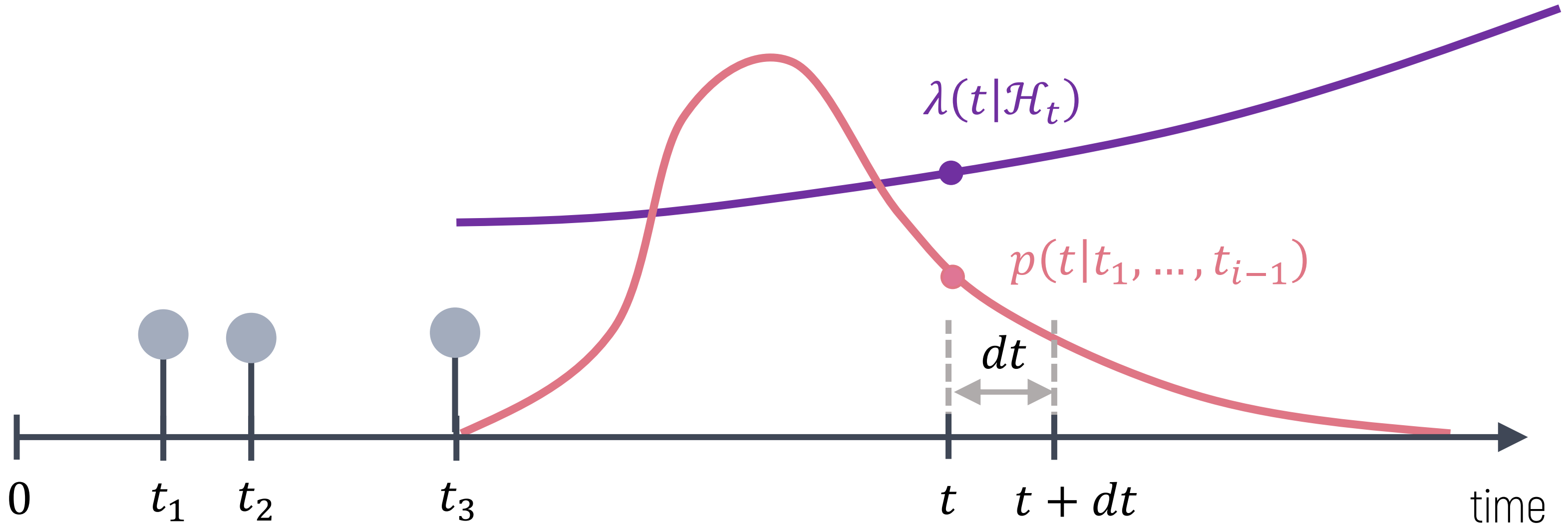
Conditional PDF vs. conditional intensity

- Conditional PDF

$$p(t|t_1, \dots, t_{i-1})dt = \Pr(\text{next event} \in [t, t + \Delta t) | \{t_1, \dots, t_{i-1}\})$$

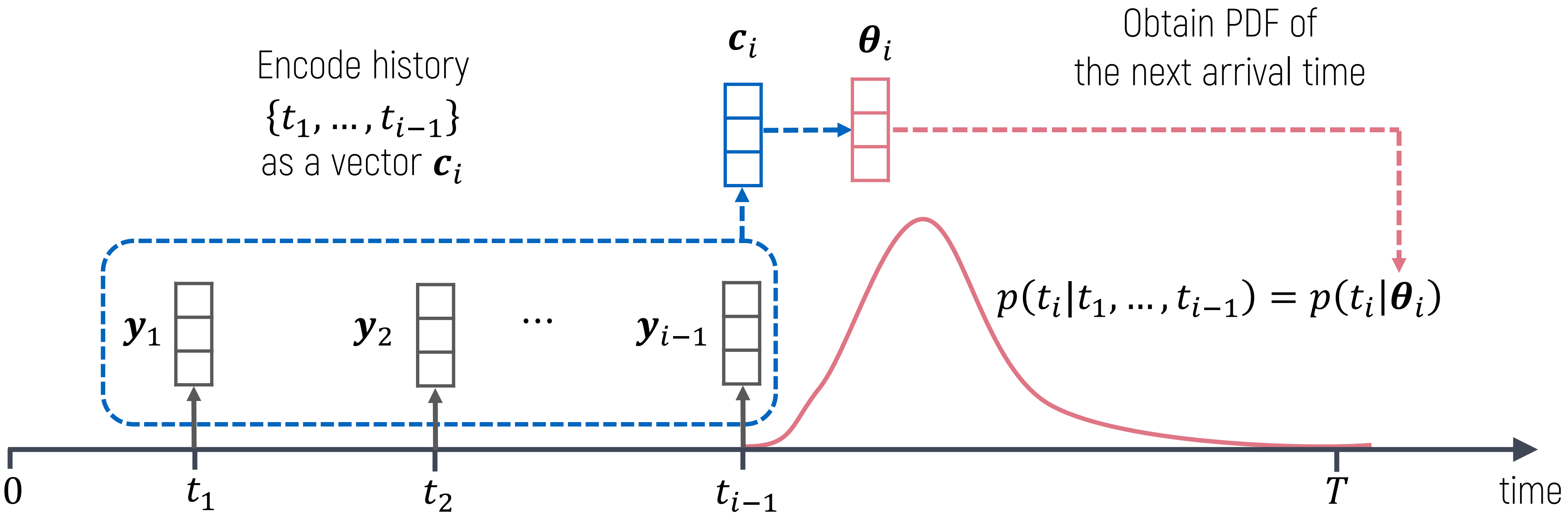
- Conditional intensity

$$\lambda(t|\mathcal{H}_t)dt = \Pr(\text{next event} \in [t, t + \Delta t) | \{\text{no event in} \in (t_{i-1}, t)\} \cup \{t_1, \dots, t_{i-1}\})$$



Autoregressive neural TPPs

- Main idea: Model the conditional PDF $p(t_i|t_1, \dots, t_{i-1})$ with neural networks

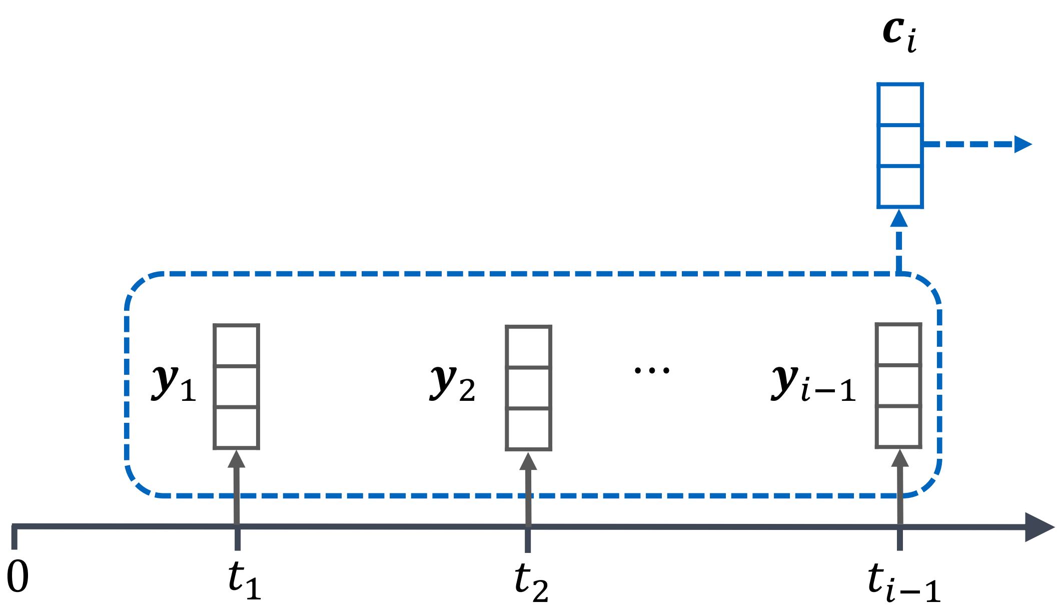


[Du, Dai, Trivedi, Upadhyay, Gomez-Rodriguez, Song, KDD 2016; Shchur, Biloš, Günnemann, ICLR 2021]

Encoding the history into a vector

1. Representing events as feature vectors $\boldsymbol{y}_i$
   - Inter-event time as feature
   - Continuous-time positional encoding

2. Aggregate feature vectors $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_{i-1}\}$ into a history embedding $\boldsymbol{c}_i$
   - RNN
   - Transformers



Modeling the conditional distribution

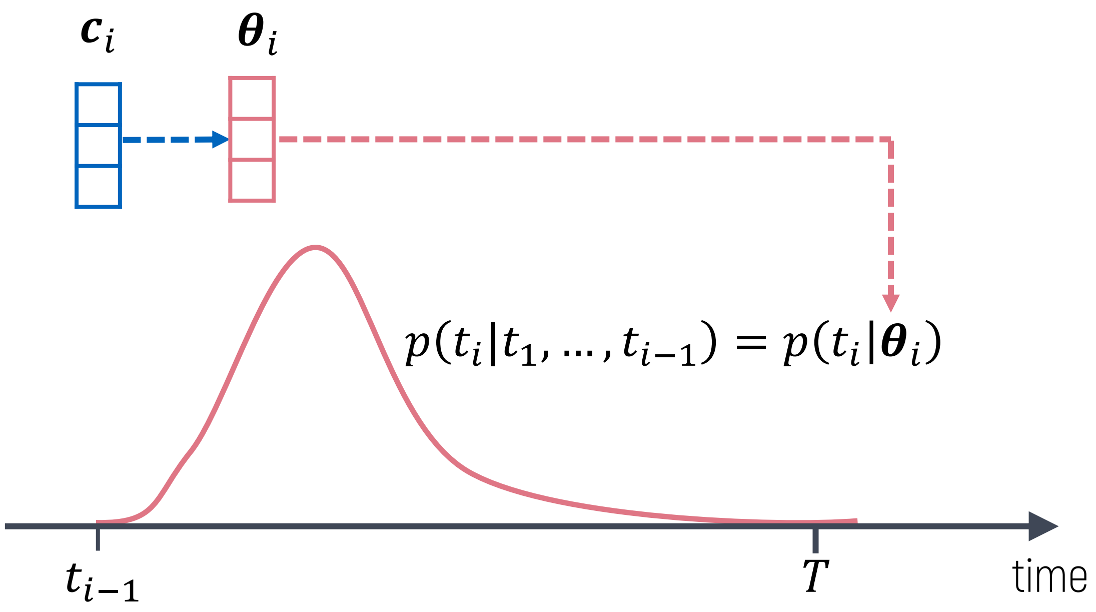
1. Pick a parametric PDF $p(\cdot\,|\boldsymbol{\theta})$ over $[0, \infty)$
   - Simple distribution (Gamma, log-normal, …)
   - Mixture distribution
   - Normalizing flows

2. Compute parameters from the history embedding

$$\boldsymbol{\theta}_i = \sigma(\boldsymbol{W}\boldsymbol{c}_i + \boldsymbol{b})$$

3. Obtain the conditional distribution

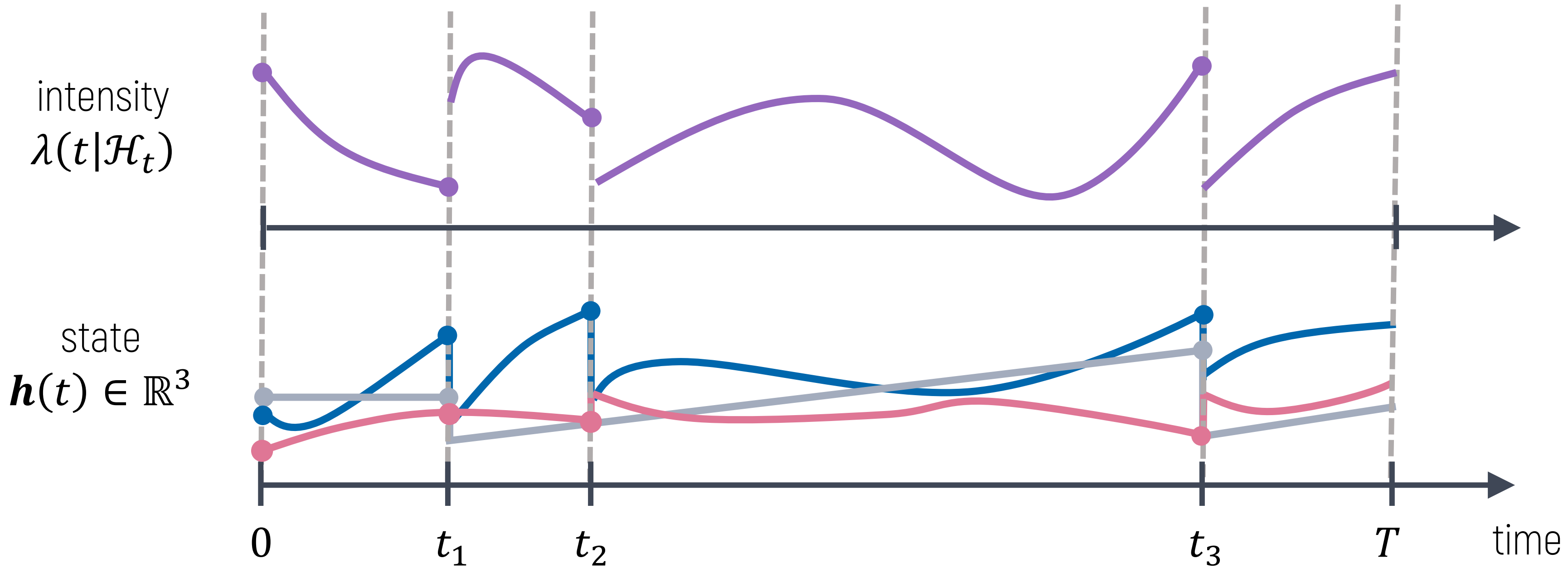
$$p(t_i | t_1, \dots, t_i) = p(t_i - t_{i-1} | \boldsymbol{\theta}_i)$$



Neural TPPs:
Continuous-time state evolution

Continuous-time state evolution

- State $\boldsymbol{h}(t)$ evolves in continuous time
- State directly defines the intensity, e.g., $\lambda(t|\mathcal{H}_t) = \exp\left(\boldsymbol{w}^T \boldsymbol{h}(t)\right)$



[Mei, Eisner, NeurIPS 2017; Jia, Benson, NeurIPS 2019; Rubanova, Chen, Duvenaud, NeurIPS 2019]

State evolution

1. Start with initial state $\boldsymbol{h}(0) \in \mathbb{R}^D$

2. State evolves continuously between events

$$\boldsymbol{h}(t+\Delta t) = \text{Evolve}(\boldsymbol{h}(t), t, t+\Delta t)$$

e.g., neural ODE

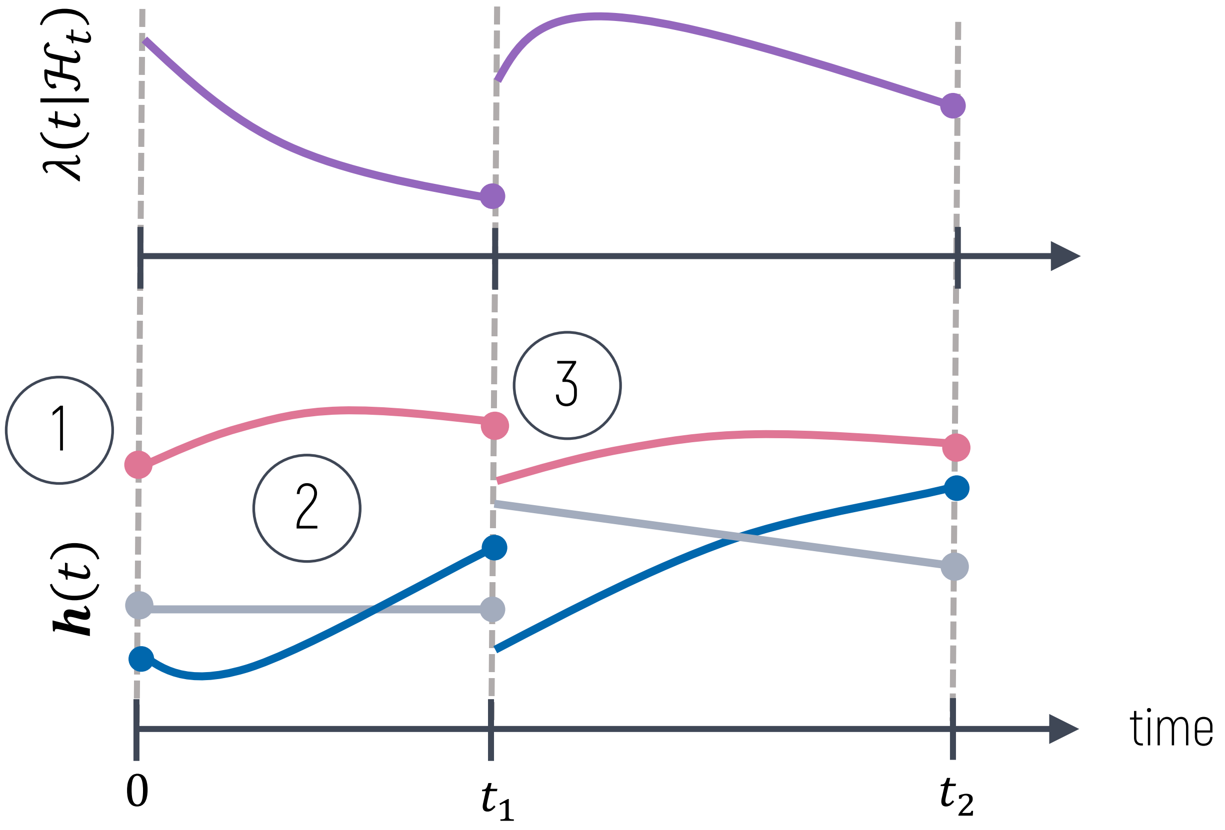
$$\boldsymbol{h}(t+\Delta t) = \boldsymbol{h}(t) + \int_t^{t+\Delta t} \frac{\partial \boldsymbol{h}(u)}{\partial u} du$$

3. Discrete update after each event

$$\boldsymbol{h}(t_i^+) = \text{Update}(\boldsymbol{h}(t_i), \boldsymbol{y}_i)$$

e.g., RNN update

$$\boldsymbol{h}(t_i^+) = \tanh(\boldsymbol{W}\boldsymbol{h}(t_i) + \boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{b})$$



Autoregressive vs. continuous-time TPPs

- Autoregressive
  - ✔ Closed-form likelihood evaluation
  - ✔ Closed-form sampling

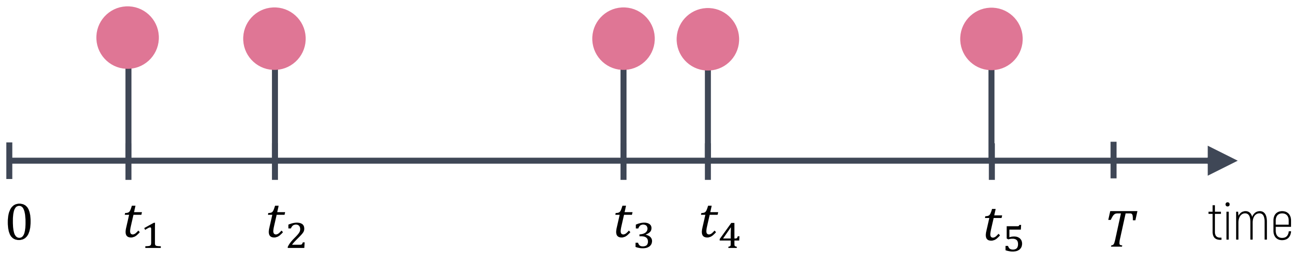
- Continuous-time state evolution
  - ✔ Naturally handle missing data
  - ✘ Require numerical integration for sampling and training

Parameter estimation

Learning: Maximum likelihood (MLE)

- Log-likelihood function

$$\log p_{\boldsymbol{\theta}}(\{t_1, \ldots, t_N\}) = \underbrace{\sum_{i=1}^{N} \log \lambda\left(t_i \mid \mathcal{H}_{t_i}\right)}_{\substack{\text{Probability of events at} \\ \text{times } \{t_1, \ldots, t_N\}}} - \underbrace{\int_0^T \lambda(u \mid \mathcal{H}_u) du}_{\substack{\text{Probability of NO events} \\ \text{in the rest of the interval}}}$$



Learning: Sampling-based losses

- Compute loss based on sampled trajectories

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\{t_1,\dots,t_N\}\sim\mathrm{TPP}_{\boldsymbol{\theta}}}[f(\{t_1,\dots,t_N\})]$$

- Equivalent to the reparametrization trick – but now for TPPs

Application	TPP_θ	$f(\{t_1,\dots,t_N\})$
Generative modeling	Learned model	Sample quality
Reinforcement learning	Policy	Reward function
Variational inference	Approximate posterior	Evidence lower bound

[Yan, Liu, Shi, Li, Zha, IJCAI 2018; Upadhyay, De, Gomez-Rodriguez, NeurIPS 2018; Shchur, Gao, Biloš, Günnemann, NeurIPS 2020]

Summary

- TPPs – probabilistic models for continuous-time event data
- Two equivalent ways to define a TPP: conditional intensity or conditional PDFs
- Neural TPPs – flexible alternatives to conventional models
- Lots of existing applications – many more to discover

Oleksandr Shchur
TU München
shchur.github.io