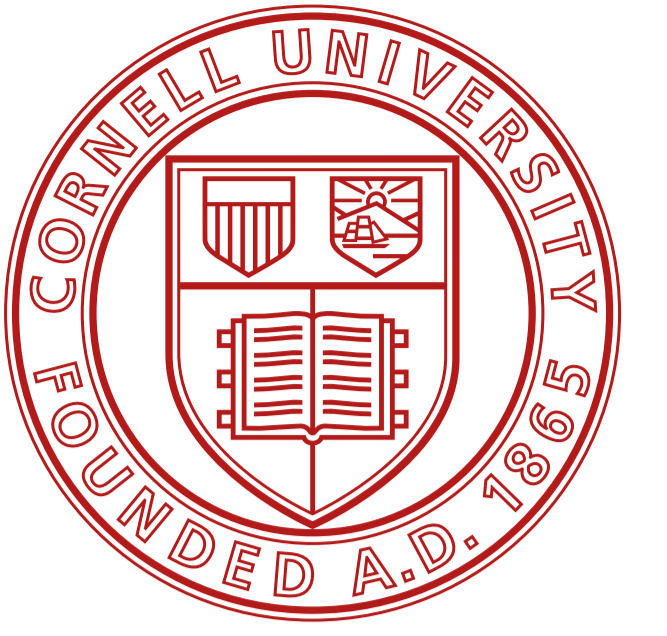


Neural Jump Stochastic Differential Equations

Junteng Jia and Austin R. Benson · Cornell University

✉ jj585@cornell.edu, arb@cs.cornell.edu

🌐 <https://github.com/000Justin000/torchdiffeq/tree/jj585>



Motivation & Problem Statement

Many real-world systems evolve continuously over time but are interrupted by stochastic events. For example, a social network user might have some evolving interest in a product that is abruptly changed by seeing an ad. How can we simultaneously learn continuous and discrete dynamics?

Given:

- $\mathcal{H}_t = \{(\tau_j, \mathbf{k}_j)\}_{\tau_j < t}$ — events up to time t ; τ_j is a timestamp and \mathbf{k}_j is an (optional) discrete or continuous label

Goal:

- learn the latent dynamics that generated \mathcal{H}_t
- predict the likelihood or label of future events

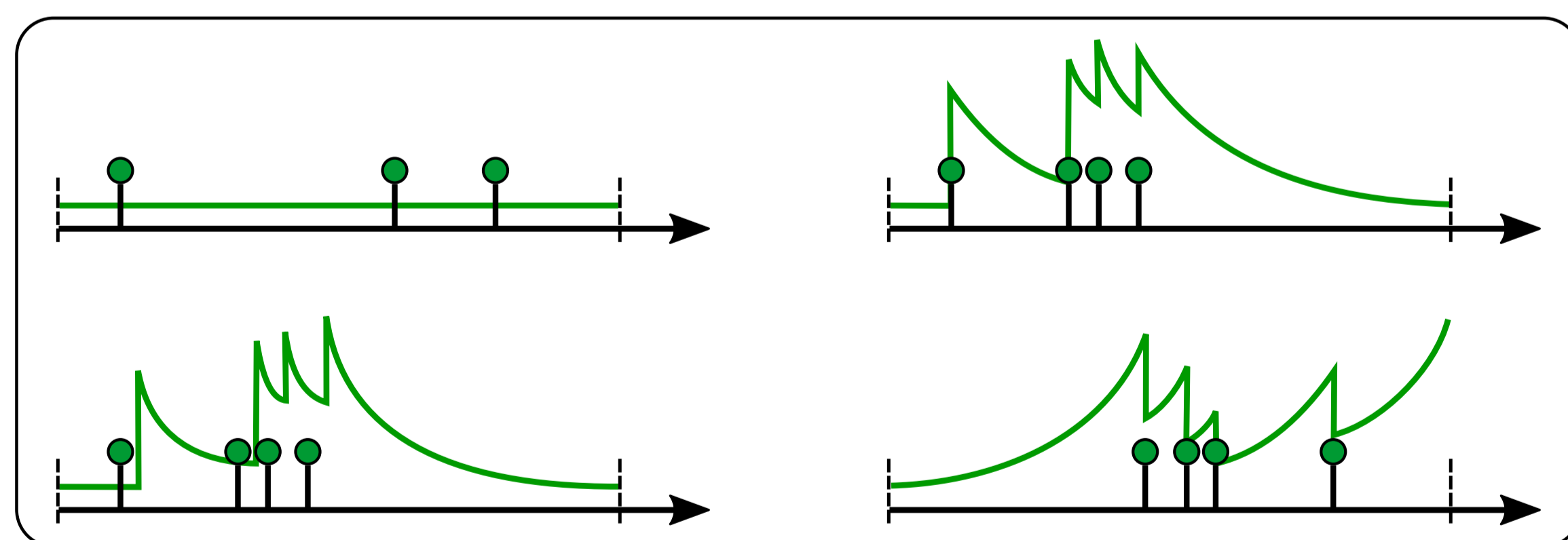
Background on Point Process Models

We model event sequences with point processes, where event generation is described by a conditional intensity:

$$\mathbb{P}\{\text{event in } [t, t + dt] \mid \mathcal{H}_t\} = \lambda(t) \cdot dt$$

Intensity dynamics depend on \mathcal{H}_t and can be written as a jump SDE. If $N(t)$ counts the number of events before t :

$$d\lambda(t) = \beta \cdot [\lambda(t) - \lambda_0] \cdot dt + \alpha \cdot dN(t)$$



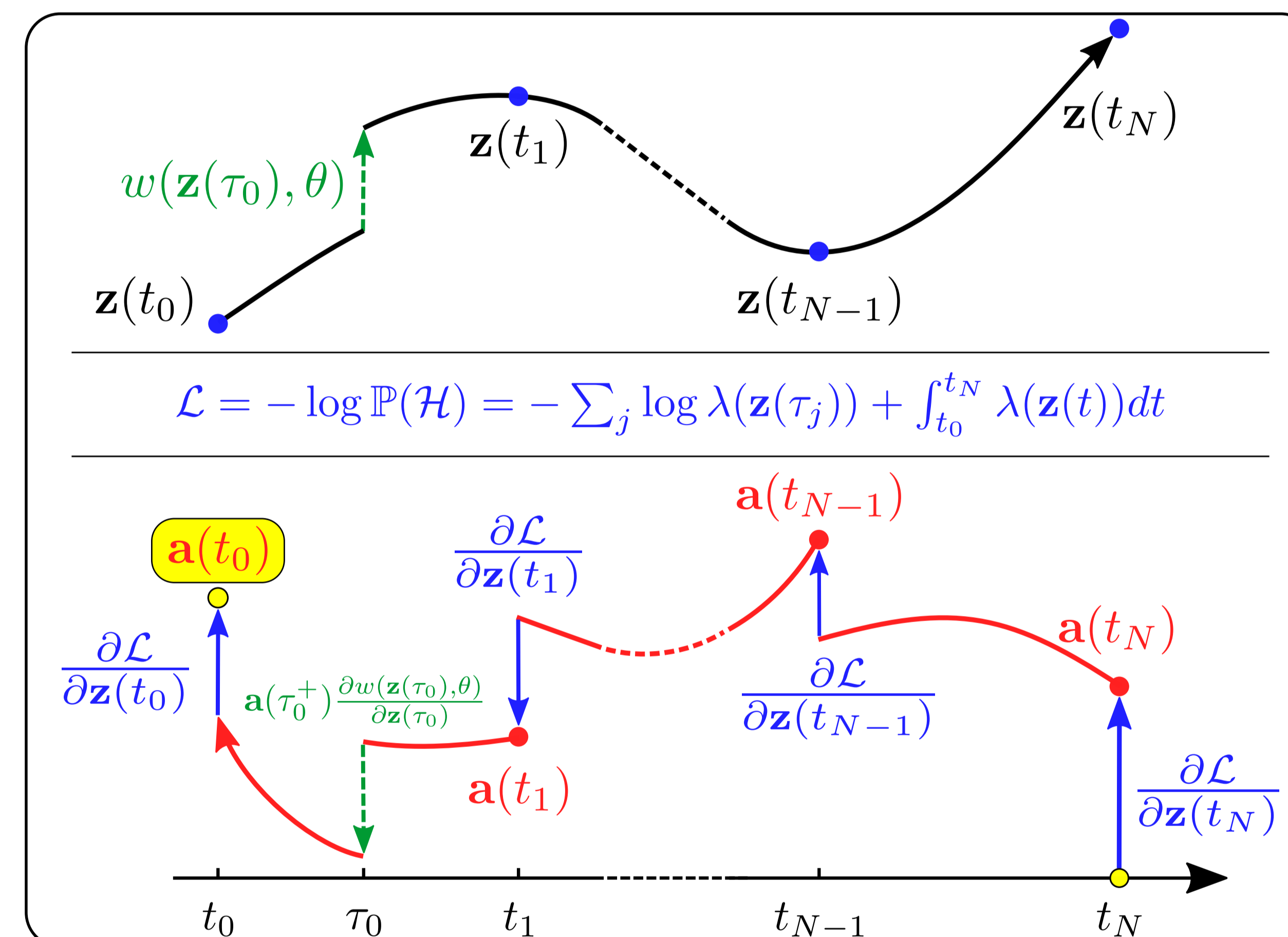
Limitation: the functional form of $\lambda(t)$ dynamics must be provided. Some widely-used function forms shown above.

Model and Learning

We follow the ideas of Neural ODEs¹ and parameterize the jump SDE model with neural nets and a latent $\mathbf{z}(t)$. This gives our neural jump SDE model (NJSDE):

$$\begin{aligned} d\mathbf{z}(t) &= f(\mathbf{z}(t), \theta) \cdot dt + w(\mathbf{z}(t), \theta) \cdot dN(t) \\ \lambda(t) &= \lambda(\mathbf{z}(t), \theta) \end{aligned}$$

We can use learned latent continuous dynamics $\mathbf{z}(t)$ for simulation and prediction.



Training with the adjoint method^{1,4}

(here just to compute the gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}(t_0)} = \mathbf{a}(t_0)$)

1. for desired loss or likelihood \mathcal{L} , set $\mathbf{a}(t_N) = \frac{\partial \mathcal{L}}{\partial \mathbf{z}(t_N)}$
2. integrate $\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t) \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)}$ backwards until event at τ
3. update $\mathbf{a}(\tau) = \mathbf{a}(\tau^+) + \mathbf{a}(\tau^+) \frac{\partial w(\mathbf{z}(\tau), \theta)}{\partial \mathbf{z}(\tau)}$
4. go to step 2

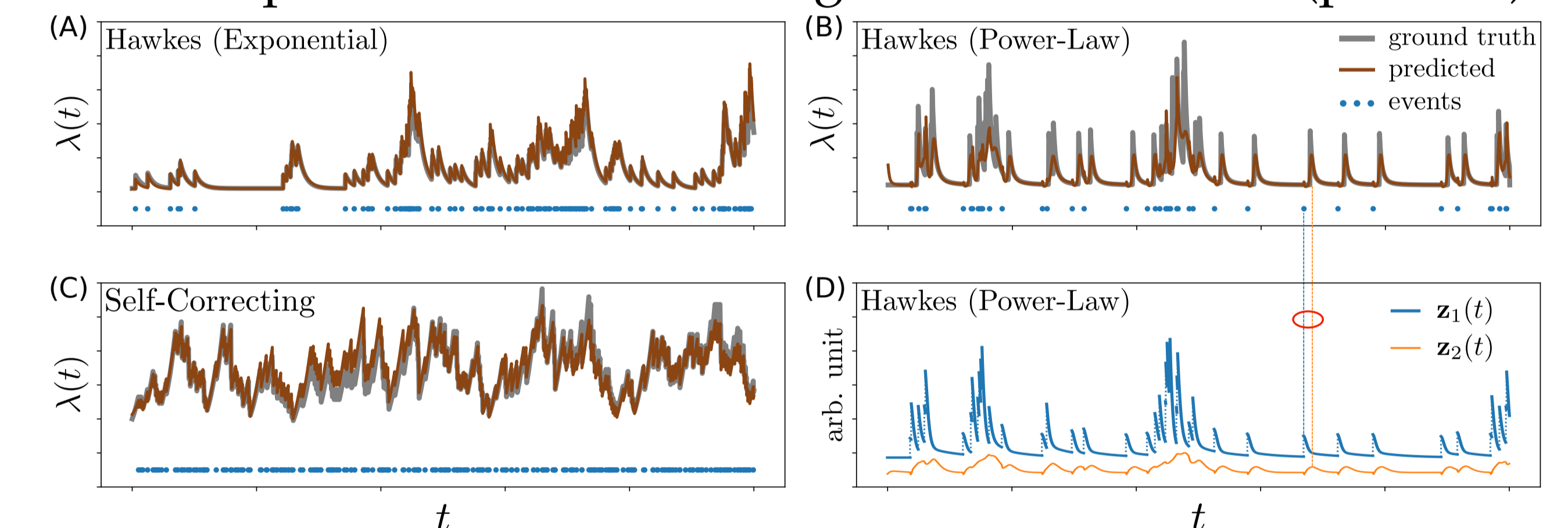
By augmenting $\mathbf{z}(t)$ to include θ , this method can be used to learn all of the latent dynamics. (See paper for details.)

Learning true conditional intensities

- Input: event sequences from classical point processes
- Output: accurately learned conditional intensities $\lambda(t)$, as measured by mean absolute percentage error (MAPE)

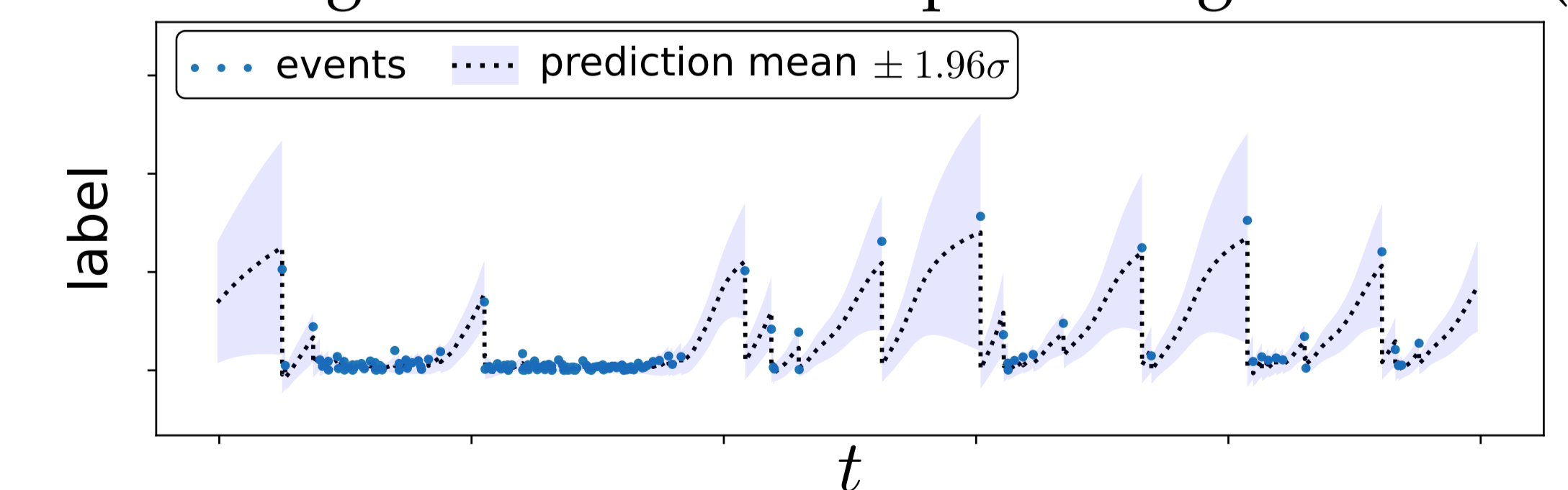
	MAPE	Hawkes (E)	Hawkes (PL)	Self-Correcting
Hawkes (E)		3.5	155.4	29.1
Hawkes (PL)		128.5	9.8	29.1
Self-Correcting		101.0	87.1	1.6
RNN		22.0	20.1	24.3
NJSDE		5.9	17.1	9.3

The NJSDE can learn complex delaying effect of power-law Hawkes process with interacting latent dimensions (panel D).



Predicting continuous outcomes (synthetic)

Event labels are sampled from a distribution $\mathbf{k} \sim p(\mathbf{k} | \mathbf{z}(t), \theta)$. Our model can predict labels with mean absolute error 0.35, an order of magnitude lower than predicting the mean (3.65).



Predicting discrete outcomes (Web / medical data)

Each event sequence is the awards history of a Stack Overflow user or the clinical visit history of a patient. The goal is to predict the award type or visiting reason for each event.

	Error Rate	[2]	[3]	NJSDE
Stack Overf ow		54.1	53.7	52.7
MIMIC2		18.8	16.8	19.8

[1] Chen et al., Neural ordinary differential equations, NeurIPS (2018).

[2] Du et al., Embedding event history to vector, KDD (2016).

[3] Mei and Eisner, The neural Hawkes process, NeurIPS (2017).

[4] Corner et al., Adjoint Sensitivity Analysis of Hybrid Multibody Dynamical Systems, arXiv (2018).