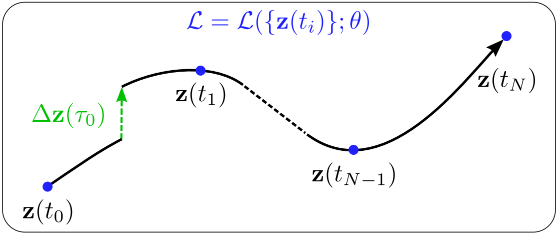
Neural Jump Stochastic Differential Equation

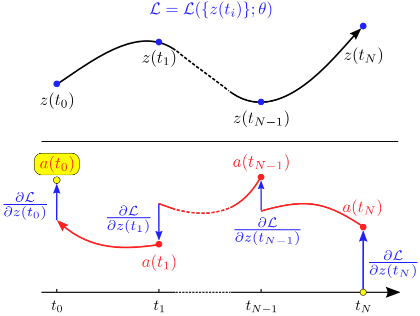


Junteng Jia and Austin R. Benson

Department of Computer Science, Cornell University, Ithaca, NY 14853

Neural Ordinary Differential Equation

Continuous-time Model for Learning Dynamical Systems



parametrize dynamics with neural net

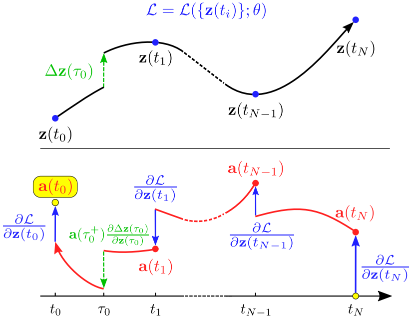
- modeling: integrate $dz(t) = f(z(t), \theta) \cdot dt$
- learning: integrate $da(t) = -a(t)\frac{\partial f(z(t),\theta)}{\partial z(t)} \cdot dt$ backwards in time

- pros:
  - $\mathcal{O}(1)$ memory learning
  - adaptive step size computation
  - continuous time-series model
- cons:
  - cannot model "jumps" in dynamics
- contributions of <u>our</u> work:
  - introduce a stochastic process term that describes "jumps"
  - derive how to handle discontinuities during back propagation
  - apply our framework to model point process dataset

Chen, Rubanova, Bettencourt and Duvenaud; NeurIPS, 2018.

Jump Stochastic Differential Equations

Continuous-time Model for Learning Dynamical Systems



augment dynamics with stochastic jump

- let $N(t)$ counts # of events up to time t:

  $dz(t) = f(z(t), \theta) \cdot dt + w(z(t), \theta) \cdot dN(t)$

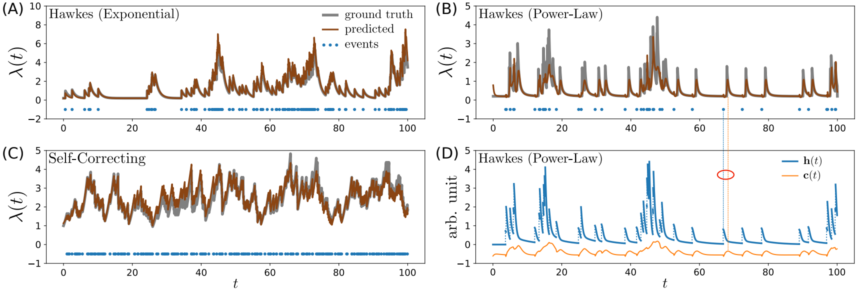
- event probability given by intensity:

  $P[N(t + dt) = k + 1 | N(t) = k] = \lambda(t) dt$

- modeling:
  - integrate $dz(t) = f(z(t), \theta) \cdot dt$ until an event happens at τ
  - update $z(\tau^+) = z(\tau) + w(z(\tau), \theta)$
  - resume integration
- learning $\frac{d\mathcal{L}}{dz(t_0)} = a(t_0)$:
  - integrate $\frac{da(t)}{dt} = -a(t)\frac{\partial f(z(t),\theta)}{\partial z(t)}$ backwards until the event at τ
  - $a(\tau) = a(\tau^+) + a(\tau^+)\frac{\partial w(z(\tau),\theta)}{\partial z(\tau)}$
  - resume integration
- remark:
  - $\mathcal{O}(1)$ memory learning maintained

Application to Time Series Analysis

Modeling Point Processes



- modeling timestamps $\{\tau_j\}$ of an event sequence
  - latent dynamics: $dz(t) = f(z(t), \theta) \cdot dt + w(z(t), \theta) \cdot dN(t)$
  - intensity parametrized with neural network $\lambda(t) = \lambda(z(t), \theta)$
  - loss: $\mathcal{L} = -\log \mathbb{P}(\{\tau_j\}) = -\sum_j \log \lambda(z(\tau_j), \theta) + \int_{t_0}^{t_N} \lambda(z(t), \theta) dt$

- model outperforms state-of-the-art (more experiments in paper)