Depp Reinforcement Learning of Marked Temporal Point Processes

Utkarsh Upadhyay
MPI-SWS
utkarshu@mpi-sws.org

Abir De
MPI-SWS
ade@mpi-sws.org

Manuel Gomez-Rodrizuez
MPI-SWS
manuelgr@mpi-sws.org

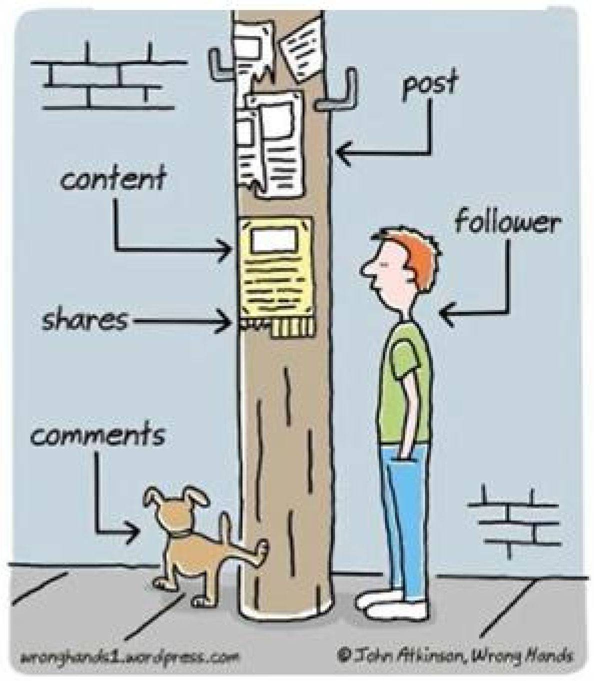
(Published in NeurIPS-2018)

孙佳佳

Outline

- **Background**
- Representation of MTPPs
- Reinforcement learning models
- Policy Optimization
- Evaluation

Discrete events in continuous time



Posts and shares

Test and diagnoses

Purchases

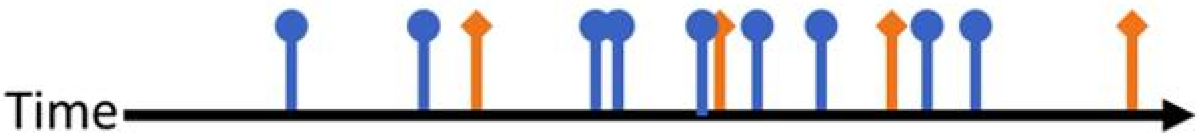
Marked temporal point processes

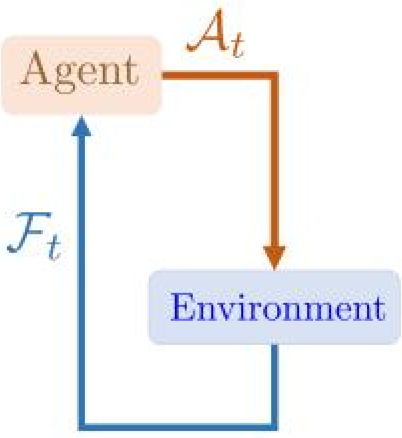
$t_i \in \mathbb{R}$: Times $\qquad z_i \in \mathbb{Z}$: Marks

$$\mathcal{H}_t = \{e_0 = (t_0, z_0), \ldots, e_n = (t_n, z_n)\}$$

- Sequence of events of type z_i with their times t_i
  - Continuous time
  - Discrete/continuous marks

What can MTPPs model?

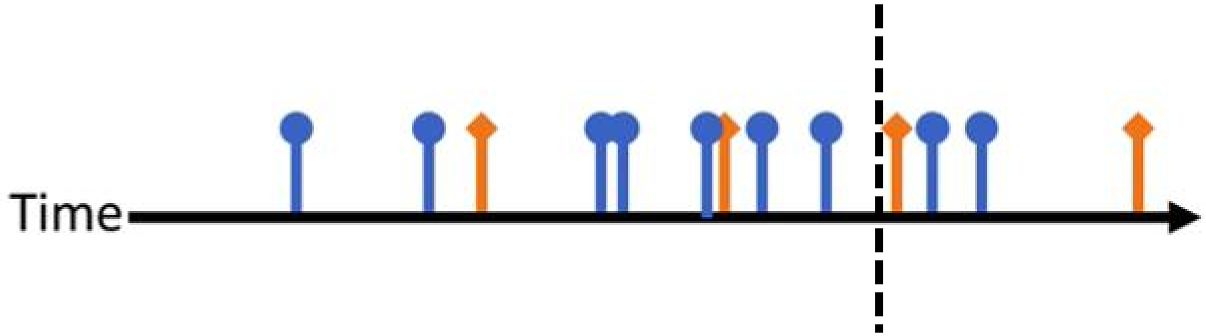


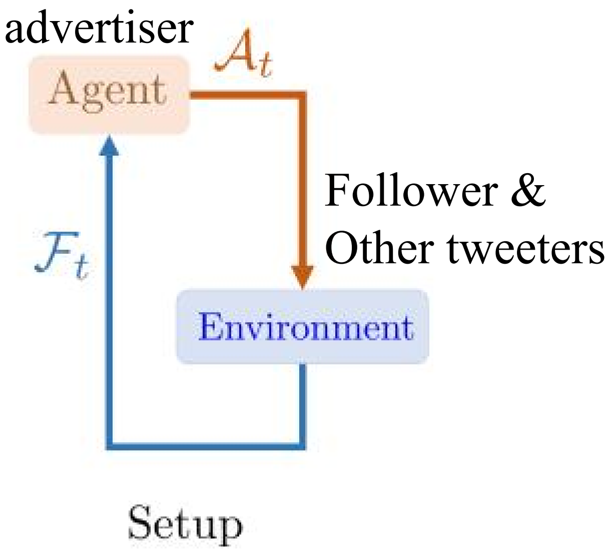


Setup

- Agent wants to maximize som reward
- Agent can perform action y_j at time t_j
- Environment provides feedback at t_i
- Reward is calculated after each episode

Example 1: When to post





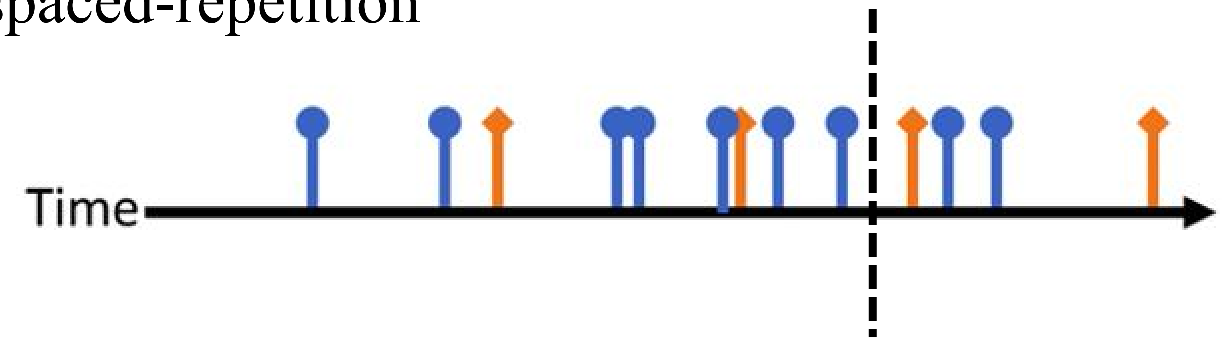
- Advertiser wants to gain attention
- Advertiser can **post tweets**
- **Other tweeter** post tweets
- **Avg.rank** is calculated at campaign end

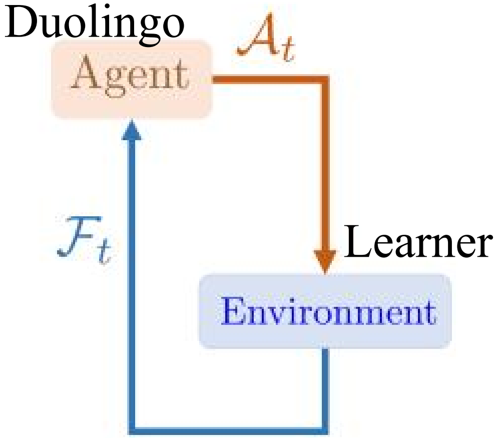
$$\mathcal{A}_t = \{9:10,\ 9:30,\ \ldots\}$$

$$\mathcal{F}_t = \{(1,\ 9:15),\ (2\ 9:31),\ \ldots\}$$

$$\boldsymbol{R}(\boldsymbol{T}) = \frac{1}{T}\int_0^T rank(t)\,dt$$

Example 2: spaced-repetition





Setup

- Duolingo wants to teach
- Duolingo can **ask questions**
- Learner can try to **recall**
- **Test score** is calculated at the end of course

$$\mathcal{A}_t = \{(abandon, 9{:}05), (reinforce, 9{:}30), \ldots\}$$

$$\mathcal{F}_t = \{(\checkmark, 9{:}05), (\times, 9{:}30), \ldots\}$$

$$\boldsymbol{R}(T) = (exam\ time)$$

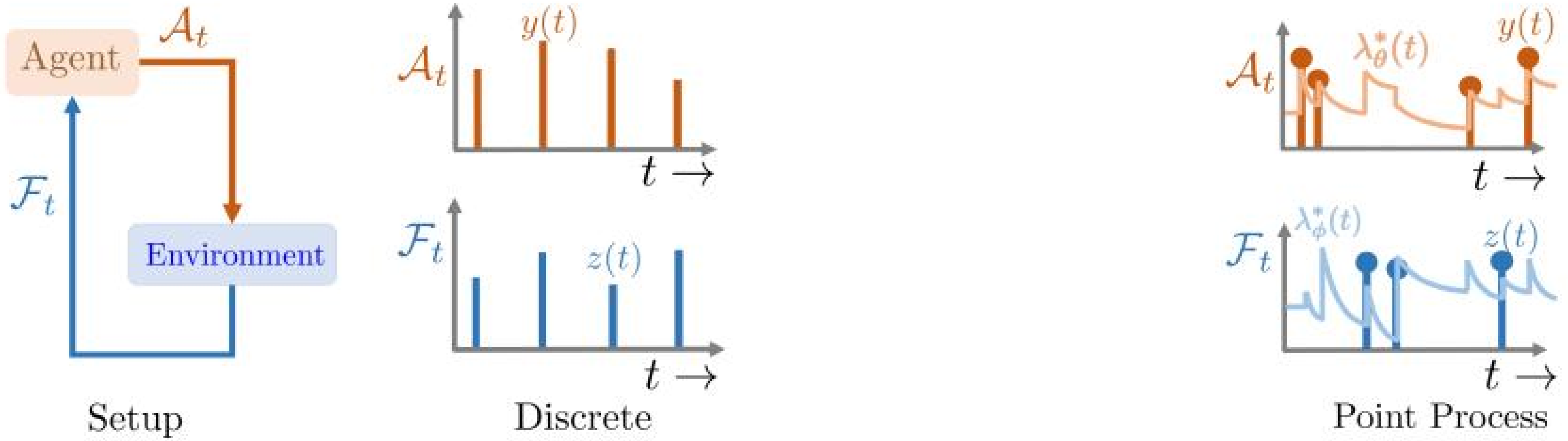
Current methods

- Rely on SDE based modeling of the **environment**
- Rely on carefully chosen **reward** functions

limitations

- **Feedback** process model is complex/unknown
- **Reward function** can be arbitrarily complex
- Cannot leverage advances in deep learning

Classical RL is too restrictive



Classical RL

- Actions and feedbacks at **discrete time steps**
- Policy determines only the mark of actions

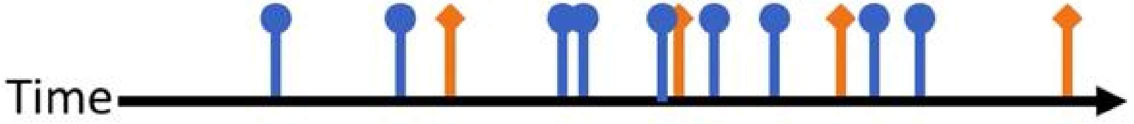
RL of MTPPs

- **Asynchronous** actions and feedbacks in continuous time
- Policy determines the time as well as mark of the next action
- **Arbitrary reward function**

Outline

- Background
- **Representation of MTPPs**
- Reinforcement learning models
- Policy Optimization
- Evaluation

How to represent MTPPs



$t_i \in \mathbb{R}$: Times $\qquad z_i \in \mathbb{Z}$: Marks

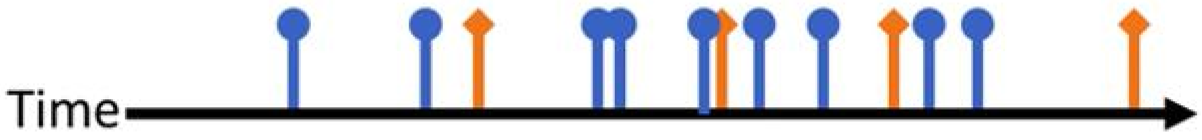
$$\mathcal{H}_t = \{e_0 = (t_0, z_0), \ldots, e_n = (t_n, z_n)\}$$

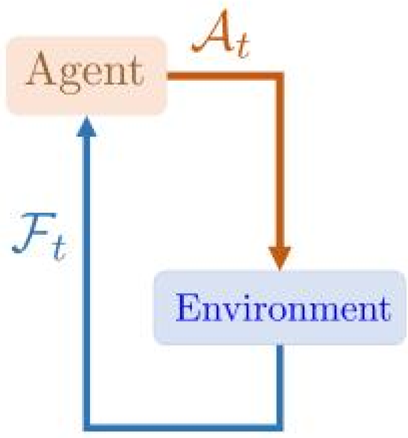
- Intensity function: $\lambda^*(t) := \mathbb{P}\{\text{event in } [t, t+dt) \mid \mathcal{H}_t\},$
- Markd distribution: $m(z \mid \mathcal{H}_t) = m^*(z)$

- Likelihood of $\mathcal{A}_T \subseteq \mathcal{H}_T$

$$\mathbb{P}(\mathcal{A}_T) := \left(\prod_{e_i \in \mathcal{A}_T} \overbrace{\lambda^*(t_i)}^{\text{Prob. of an action at } t_i} \underbrace{m^*(z_i)}_{\text{Prob. of mark } z_i} \right) \overbrace{\exp\left(-\int_0^T \lambda^*(s)\, ds \right)}^{\text{Prob. of no actions at } t \in [0,T] \setminus \{t_i\}}$$

How to represent our problem as MTPPs





Setup

- Actions: $\mathcal{A} = \{e_i = (t_i, y_i)\}$, where $(t_i, y_i) \sim p^*_{\mathcal{A};\theta} = (\lambda^*_\theta, m^*_\theta)$
- Feedbacks: $\mathcal{F} = \{f_i = (t_i, z_i)\}$, where $(t_i, z_i) \sim p^*_{\mathcal{F};\phi} = (\lambda^*_\phi, m^*_\phi)$

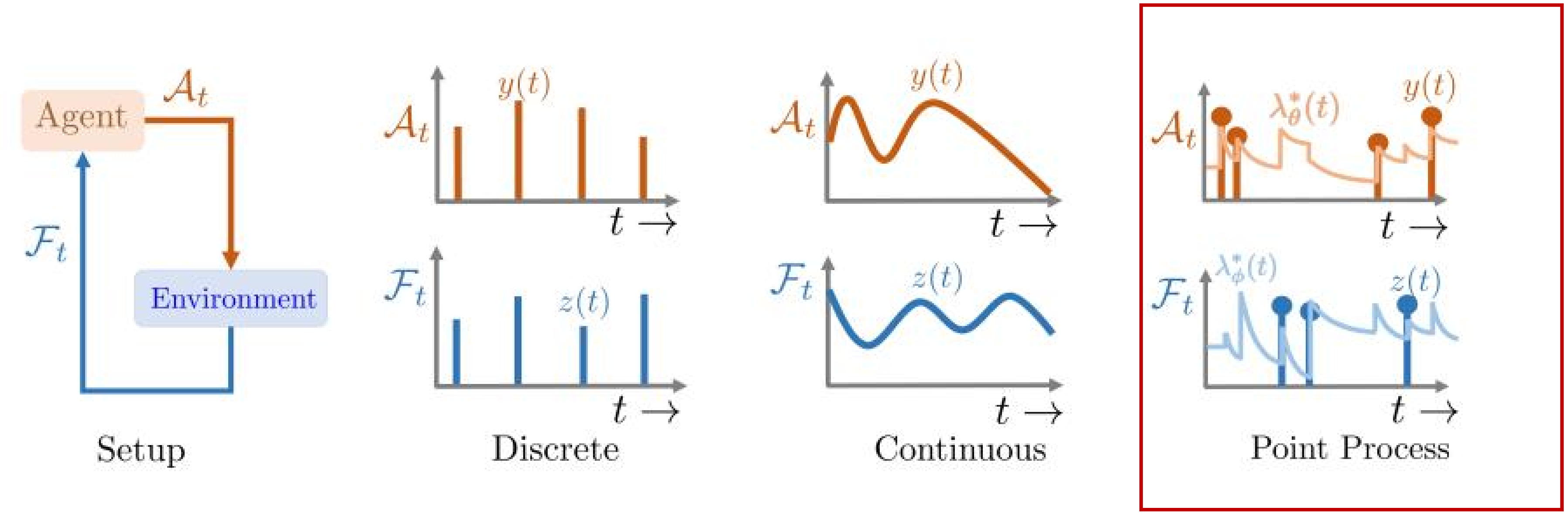
Model free reinforcement learning

$$\underset{p^*_{\mathcal{A};\theta}(\cdot)}{\text{maximize}} \quad \mathbb{E}_{\mathcal{A}_T \sim p^*_{\mathcal{A};\theta}(\cdot), \mathcal{F}_T \sim p^*_{\mathcal{F};\phi}(\cdot)} \left[R^*(T) \right]$$

Outline

- Background
- Representation of MTPPs
- **Reinforcement learning models**
- Policy Optimization
- Evaluation

Reinforcement Learning model



- Design the policy
- Realize the policy

- Actions and feedback are **asynchoronous**
- Characterized by **intensity function**

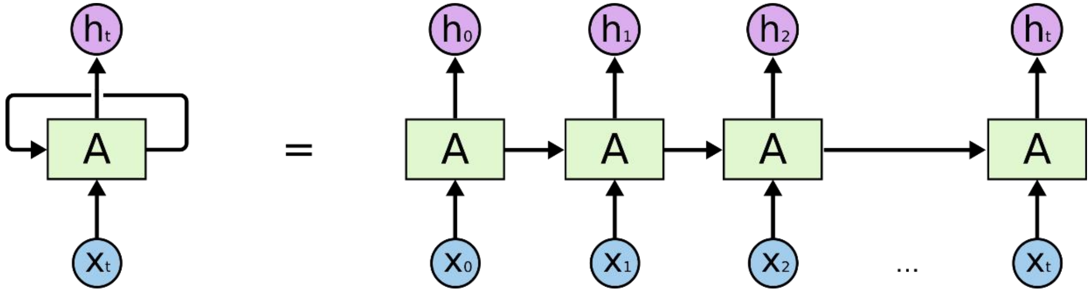
Policy parametrization

$$\lambda^*(t) := \mathbb{P}\{\text{event in } [t, t+dt) \,|\, \mathcal{H}_t\},$$

$$\mathcal{H} = \{e_0 = (t_0, z_0), e_1 = (t_1, z_1), \ldots, e_n = (t_n, z_n)\}$$

Embed the history using RNN

- Each event updates the hidden state $\mathbf{h}$
- Intensity function is determined by $\lambda_k(t) = f_k(\mathbf{w}_k^\top \mathbf{h})$



Embed the history using RNN

Input layer

$$\boldsymbol{\tau}_i = \boldsymbol{W}_t \boxed{(t_i - t_{i-1})} + \boldsymbol{b}_t,$$
$$\mathbf{b}_i = \boldsymbol{W}_a(1 - e_i) + \boldsymbol{W}_f e_i + \boldsymbol{b}_b,$$

$$\boxed{\boldsymbol{y}_i} = \boldsymbol{W}_y y_i + \boldsymbol{b}_y \quad \text{if } e_i = 0$$
$$\boxed{\boldsymbol{z}_i} = \boldsymbol{W}_z z_i + \boldsymbol{b}_z \quad \text{if } e_i = 1$$

Hidden layer

$$\boldsymbol{h}_i = \tanh(\boldsymbol{W}_h \boldsymbol{h}_{i-1} + \boldsymbol{W}_1 \boldsymbol{\tau}_i + \boldsymbol{W}_2 \boldsymbol{y}_i + \boldsymbol{W}_3 \boldsymbol{z}_i + \boldsymbol{W}_4 \mathbf{b}_i + \boldsymbol{b}_h)$$

Output layer

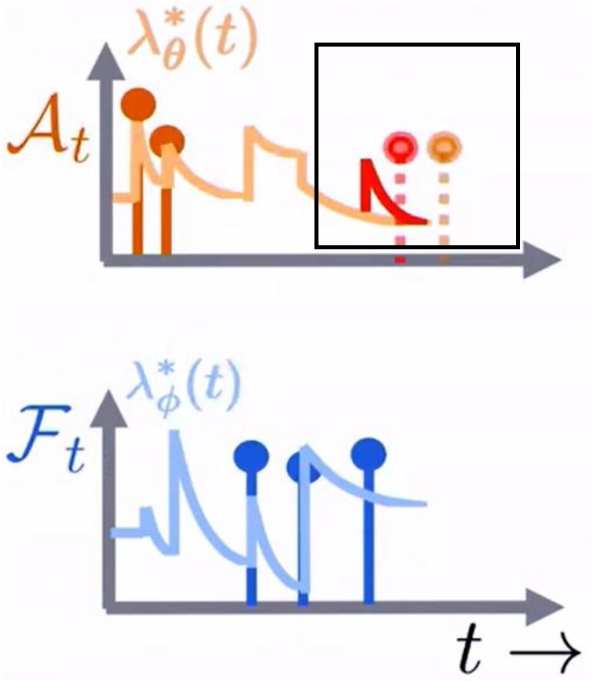
$$\lambda^*_\theta(t) = \exp\left(\boxed{b_\lambda} + \boxed{w_t}(t - t_i) + \boxed{\boldsymbol{V}_\lambda} \boldsymbol{h}_i\right)$$

$$\mathbb{P}[y_{i+1} = c] = \frac{\exp(\boldsymbol{V}^y_{c,:} \boldsymbol{h}_i)}{\sum_{l \in \mathcal{Y}} \exp(\boldsymbol{V}^y_{l,:} \boldsymbol{h}_i)}$$

$$\begin{aligned}\theta = \{&V_\lambda, V^y, w_t, b_\lambda,\\ &W_z, b_z, W_t, b_t, W_b, b_b,\\ &W_h, b_h, W_y, b_y\}\end{aligned}$$

RL with Asynchronous Feedback

- Distribution of actions may **change** because of incoming feedback



Algorithm 1: Returns the next action time

Input: Parameters $b_\lambda, w_t, \boldsymbol{V}_\lambda, \boldsymbol{h}_i$, last event time t'
Output: Next action time t
$CDF(\bullet) \leftarrow$ Cumulative distribution of next arrival time
$u \leftarrow \text{UNIF}[0, 1]$
$t \leftarrow \boxed{CDF^{-1}(u)}$
while $t < T$ **do**
  $(s, z) \leftarrow \text{WAITUNTILNEXTFEEDBACK}(t)$
  if feedback arrived before t **then**
    $CDF(\bullet) \leftarrow \text{MODIFY}(CDF(\bullet), s, z)$
    $t \leftarrow CDF^{-1}(u)$
  else
    return t
  end if
end while
return t

RL problem with PTPPs: summary

- **New**: RL problem in continuous time with discrete events

  - **Embedding state** to real vectors using RNN
  - Efficient and unbiased **sampling** procedure to handle asynchronous feedbacks

Outline

- Background
- Representation of MTPPs
- Reinforcement learning models
- **Policy Optimization**
- Evaluation

Policy Gradient method

- Maximizing the expected reward

$$J(\theta) = \mathbb{E}_{\mathcal{A}_T \sim p^*_{\mathcal{A};\theta}(\cdot), \mathcal{F}_T \sim p^*_{\mathcal{F};\phi}(\cdot)} [R^*(T)]$$

- SGD Updates:

$$\theta_{l+1} = \theta_l + \alpha_l \nabla_\theta J(\theta)|_{\theta=\theta_l}$$

- REINFORCE trick:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{A}_T \sim p^*_{\mathcal{A};\theta}(\cdot), \mathcal{F}_T \sim p^*_{\mathcal{F};\phi}(\cdot)} [\boxed{R^*(T)} \nabla_\theta \log \mathbb{P}_\theta(\mathcal{A}_T)]$$

$$\log \mathbb{P}_\theta(\mathcal{A}_T) = \sum_{e_i \in \mathcal{A}_T} (\log \lambda^*_\theta(t_i) + \log m^*_\theta(z_i)) - \int_0^T \lambda^*_\theta(s)\, ds.$$

- No need to model the feedback process
- Reward function can be arbitrary

Outline

- Background
- Representation of MTPPs
- Reinforcement learning models
- Policy Optimization
- **Evaluation**

Spaced repetition

$$\mathcal{A}_t = \{(abandon, 9{:}05), (reinforce, 9{:}30), \dots\}$$

$$\mathcal{F}_t = \{(\checkmark, 9{:}05), (\times, 9{:}30), \dots\}$$

$$\mathrm{R}(T) = (exam\ time)$$

Online learning platform

TPPRL	RL of MTPP; **No assumption** on feedback; **Arbitrarily** complex reward function
MEMORIZE	Has full access to the student model; Design **specific reward function**
Uniform	Choose items uniformly at random

(a) Recall (b) Items' difficulty

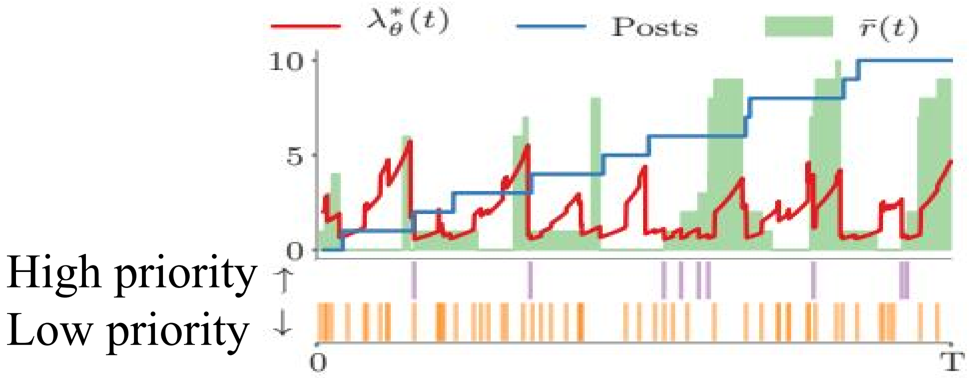
When to post

TPPRL	RL of MTPP; **No assumption** on feedback; **Arbitrarily** complex reward function
RQ	**feeds** sorted in reverse chronological order; **minimize** the average rank; intensity $\propto$ $\text{rank}_{\text{chrono}}(t)$
RQ^*	feeds sorted in reverse chronological order; minimize the average rank; intensity $\propto$ $\text{rank}_{\text{priority}}(t)$
Karimi	**feeds** sorted in reverse chronological order; **maximize** the time at the top

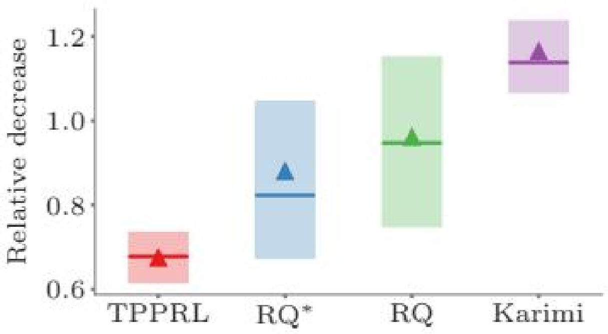
$$\mathcal{A}_t = \{9:10, 9:30, \ldots\}$$

$$\mathcal{F}_t = \{(1, 9:05), (2\ 9:30), \ldots\}$$

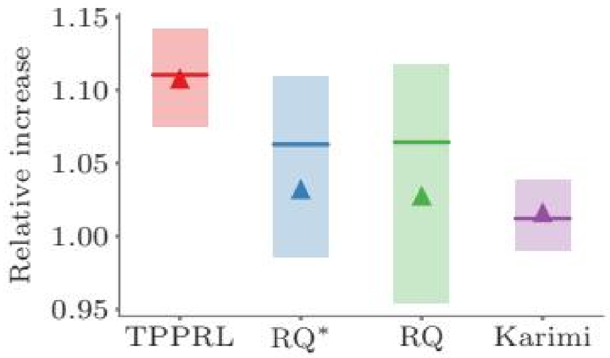
$$\mathrm{R}(T) = \frac{1}{T}\int_0^T rank(t)dt$$



(c) Example



(a) Average rank



(b) Time at top

Conclusion

Reinforcement Learning method for:

- Asynchronous actions/rewards represented as MTPPs
- Arbitrary reward functions

Funture work:

- Deriving more sophisticated reinforcement learning algorithms
- Multi-agent reinforcement learning

Thank you!