Advanced Prediction Models

Today's Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

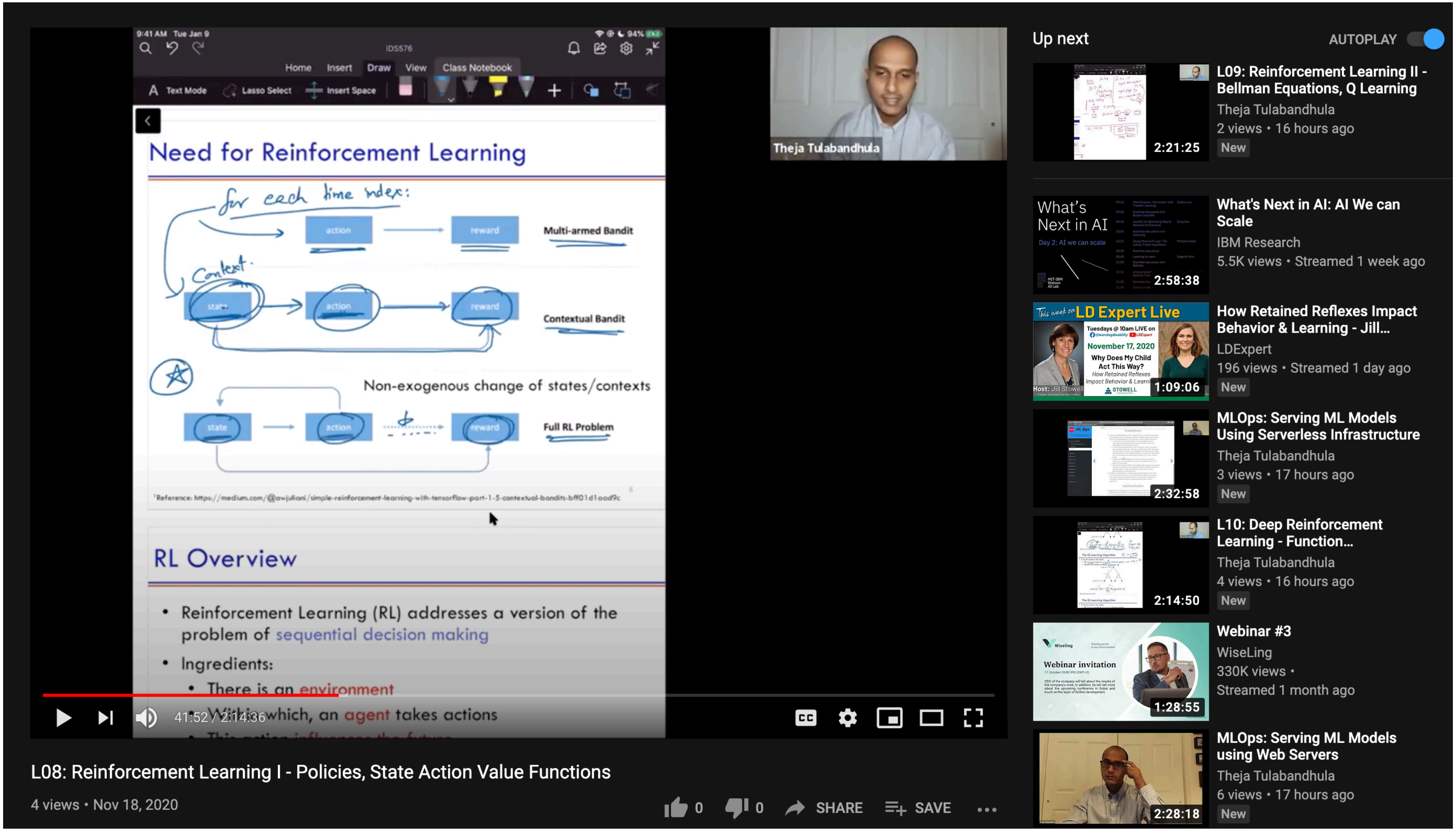
Knowledge Aware Attentive Sequential Recommendations

With Mehrnaz Amjadi and Danial Mohseni Taheri, UIC

Paper: KATRec: Knowledge Aware aTtentive Sequential Recommendations

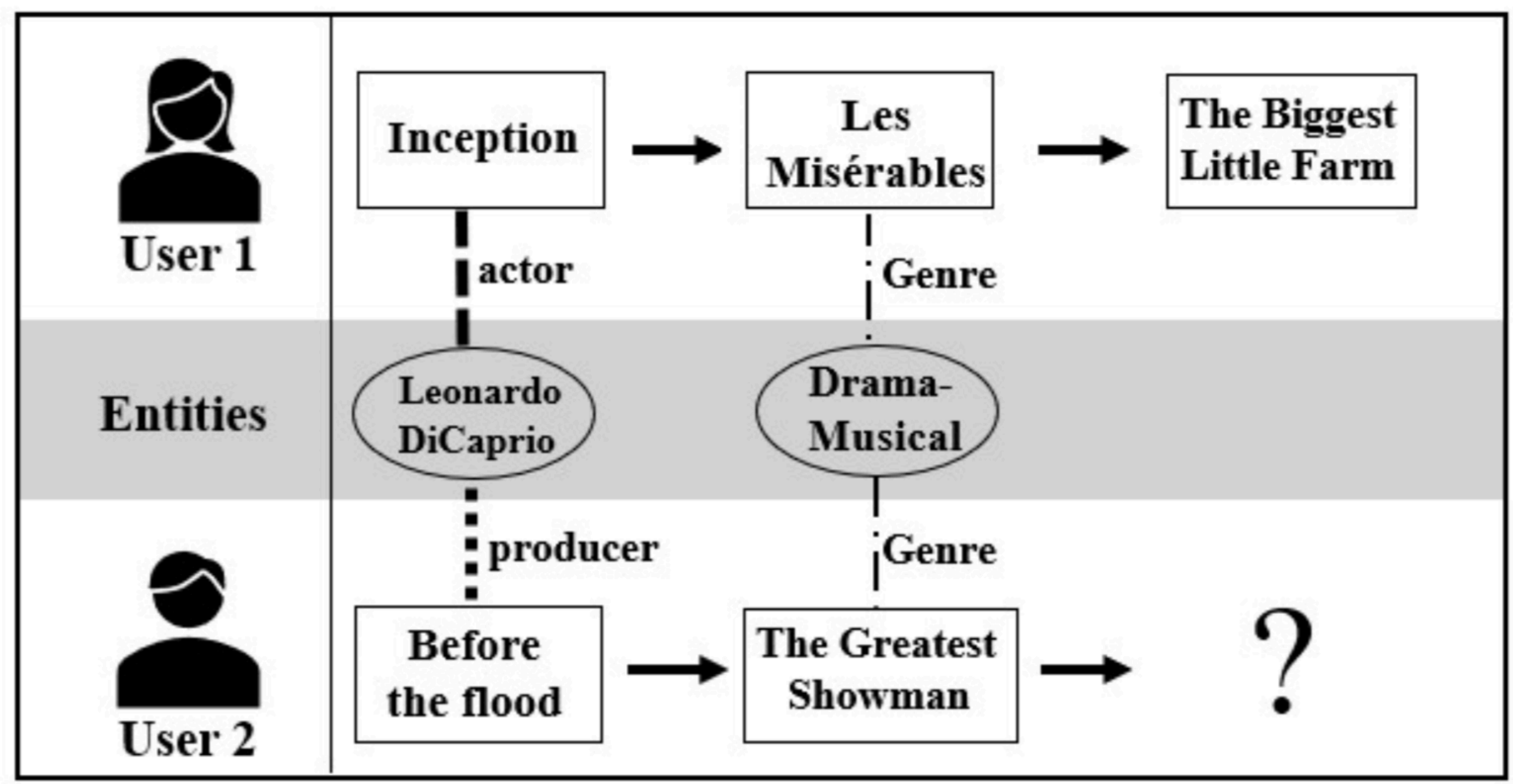
Draft Available (2020)

Knowledge Aware Attentive Sequential Recommendations



Knowledge Aware Attentive Sequential Recommendations

- Sequential recommendation systems model the dynamic preferences of users based on their historical interactions with items.
- Modeling short-term and long-term behaviour of users is challenging
- Can collaborative signal be detected via shared entities?

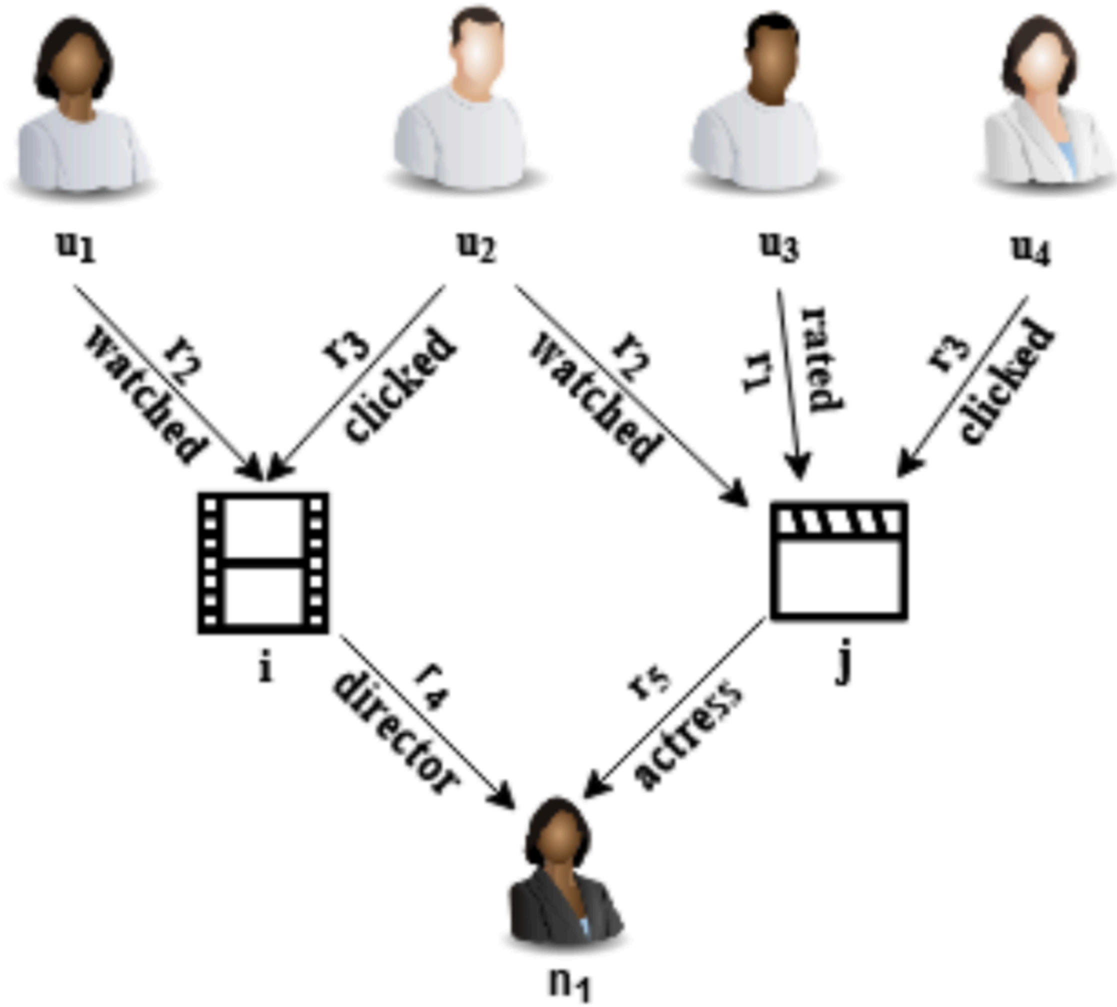


Knowledge Aware Attentive Sequential Recommendations

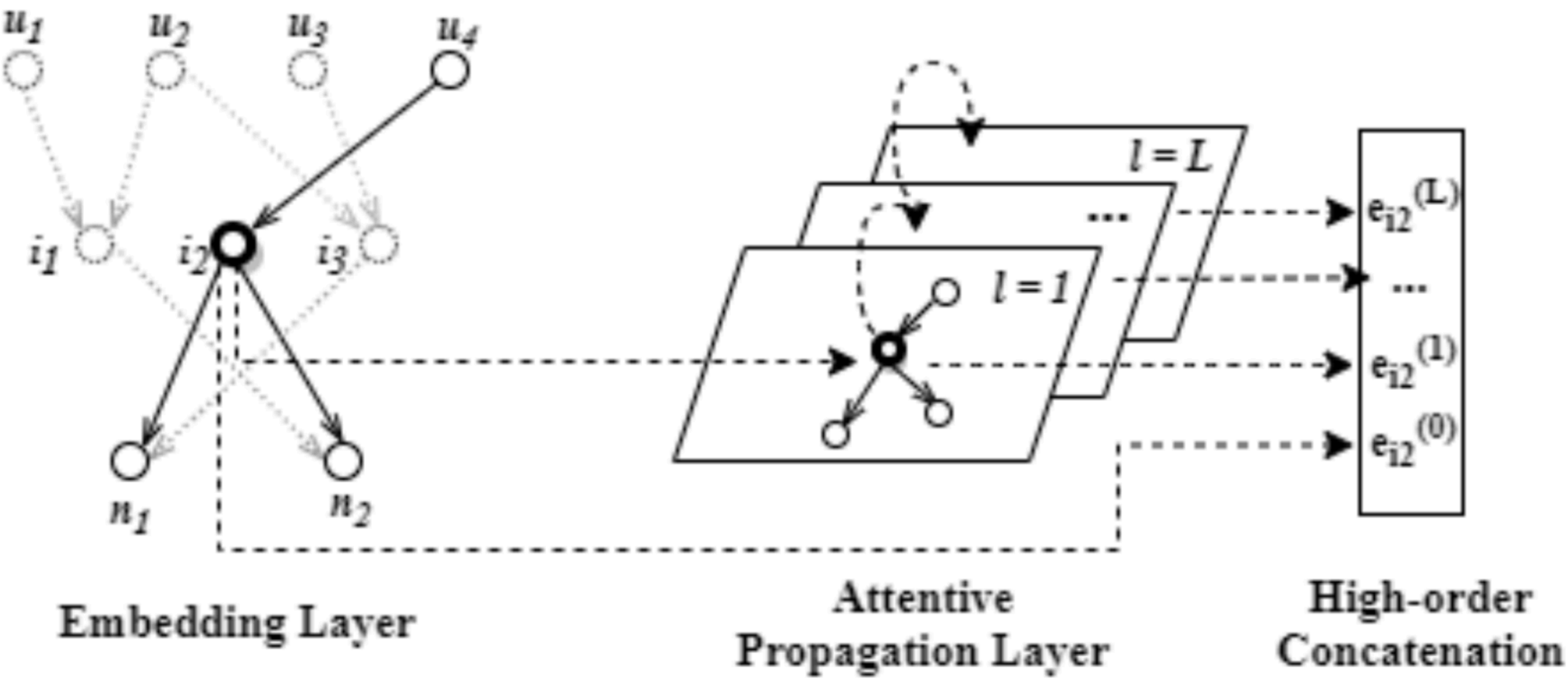
- Essentially two components in the proposed solution:
  - A bi-directional BERT like architecture that captures sequential patterns (of items) per user
  - A Knowledge graph based representation of items such that higher-order item-item relationships are adequately captured
    - Leverage pre-existing side information
    - Captures multi-relationships between items and improves their representations by considering their higher-order connectivity with neighbors on a graph

- Use both these components to make predictions of recommended items

Knowledge Aware Attentive Sequential Recommendations

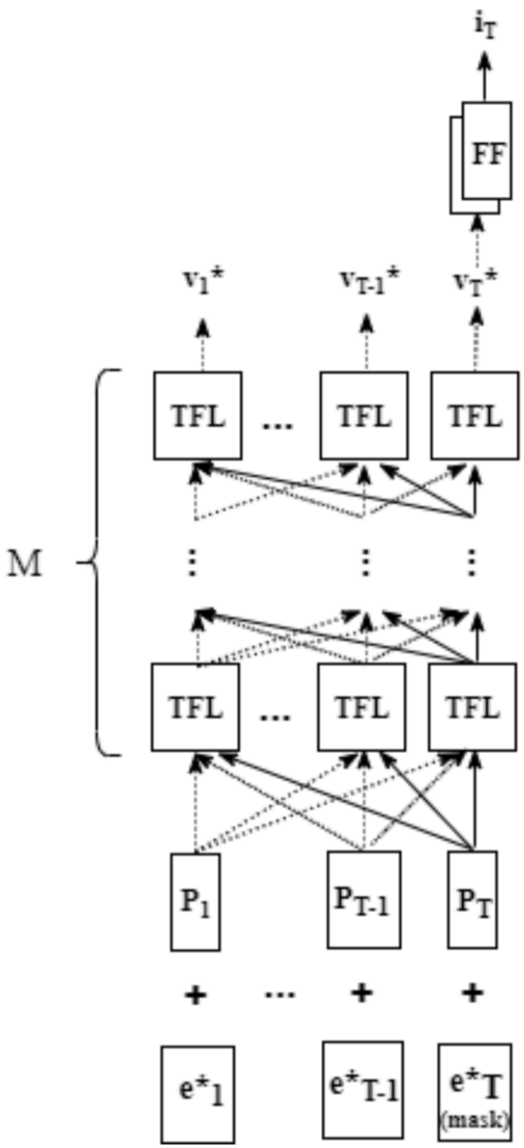


Knowledge Aware Attentive Sequential Recommendations



Knowledge Aware Attentive Sequential Recommendations

- The architecture to process the sequential information is the same as BERT.



Knowledge Aware Attentive Sequential Recommendations

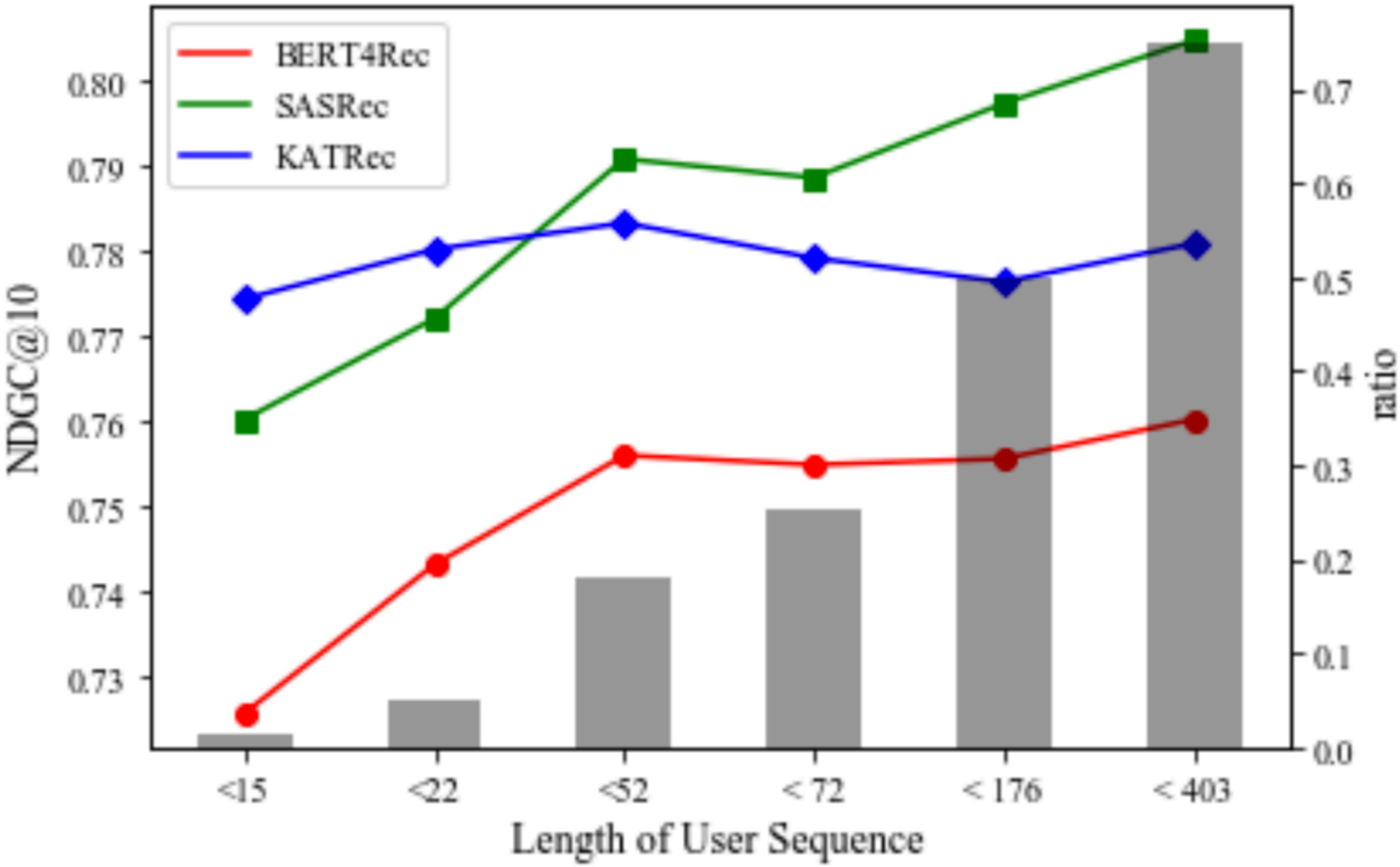
Table 1: Statistics of datasets

Datsets	Users	Items	Interactions	Entities	Relations	Triplets	Sparsity
Amazon-book	70679	24915	846434	88572	39	2557746	0.048%
LastFM	23566	48123	8057269	58266	9	464567	0.7105%
Yelp2018	45919	45538	1185068	90961	42	1853704	0.057%

Knowledge Aware Attentive Sequential Recommendations

Datasets	Metrics	GRU	GRU^{++}	SASRec	BERT	KATRec	Improv.
Amazon	NDCG@1	0.3485	0.3464	0.3749	0.4344	**0.4706**	8.33%
	NDCG@5	0.4404	0.4358	0.5267	0.5715	**0.6110**	6.91%
	NDCG@10	0.4598	0.4574	0.5600	0.6022	**0.6401**	6.2%
	Hit@5	0.5202	0.5148	0.6594	0.6910	**0.7321**	5.94%
	Hit@10	0.58	0.5814	0.7621	0.7856	**0.8217**	4.6%
	MAP	0.42	0.4259	0.5065	0.5539	**0.5907**	6.64%
LastFM	NDCG@1	0.3646	0.3523	0.6771	0.6339	**0.6931**	2.36%
	NDCG@5	0.4648	0.4448	**0.7765**	0.7606	0.7725	-0.51%
	NDCG@10	0.4881	0.4674	**0.7930**	0.7786	0.7911	-0.24%
	Hit@5	0.5531	0.5263	**0.8600**	0.8281	0.8426	-2.06%
	Hit@10	0.6249	0.5958	**0.9105**	0.8836	0.9001	-1.15%
	MAP	0.4577	0.4357	0.7598	0.7509	**0.7618**	0.26%
Yelp2018	NDCG@1	0.3946	0.4148	0.3723	0.4149	**0.4405**	6.17%
	NDCG@5	0.5041	0.5143	0.5703	0.6039	**0.629**	4.15%
	NDCG@10	0.5278	0.5395	0.6068	0.6400	**0.663**	3.6%
	Hit@5	0.5991	0.6021	0.7434	0.7690	**0.7927**	3.08%
	Hit@10	0.6721	0.68	0.8551	0.8796	**0.899**	2.2%
	MAP	0.49	0.515	0.5351	0.5706	**0.5946**	4.21%

Knowledge Aware Attentive Sequential Recommendations



Questions?

Today’s Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

User Engagement Model based on Choices

With Saketh Karra, UIC

Paper: Choice-Aware User Engagement Modeling on Social Media

Draft Available (2020)

Presented at INFORMS 2020

Engagement on Twitter

- The amount of information/content is overwhelming
- Tweets may not be consistently interesting
- Goal: maximize user engagement with content (in the form of like, reply, retweet, and retweet with comments) on the Twitter platform.
- Formulate the engagement forecasting task as a multi-label classification problem
- It captures choice behavior on an unsupervised clustering of tweet topics
- The deep neural network incorporates recent user engagement history and predicts choice conditional on this context
- Solve a tweet optimization problem based on this
- Use a large dataset obtained from Twitter

User Engagement Model based on Choices

Figure: A tweet from Elon Musk [Belkacem et al., 2019]

- On the *home timeline*, tweets can appear in an algorithmically ranked order or in the reverse chronological order, depending on pre-defined user preferences
- Users can engage with the tweet in the form of like, Retweet, or reply to the tweets or Retweet with a comment

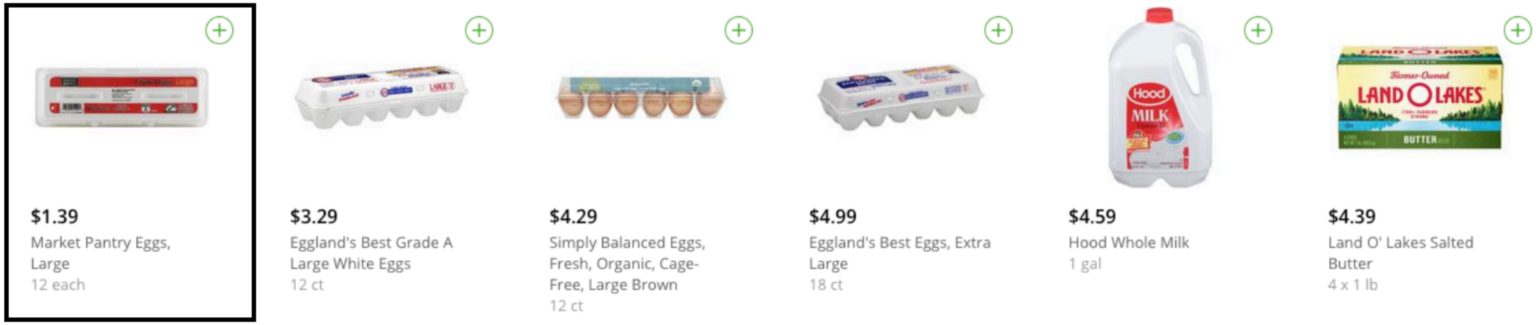
User Engagement Model based on Choices

jda. | Category Management

NIELSEN
ASSORTMENT &
SPACE OPTIMISATION
FASTEST ▸ MOST ACCURATE ▸ GROWTH

4R

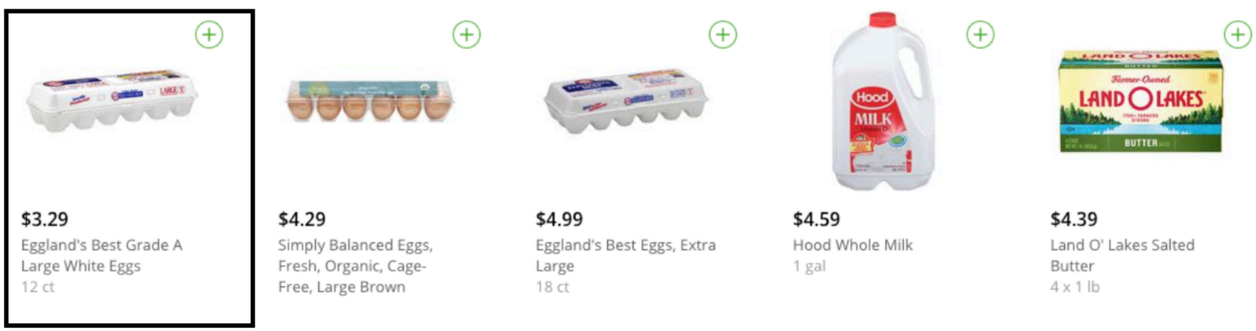
User Engagement Model based on Choices



Type to enter a caption.

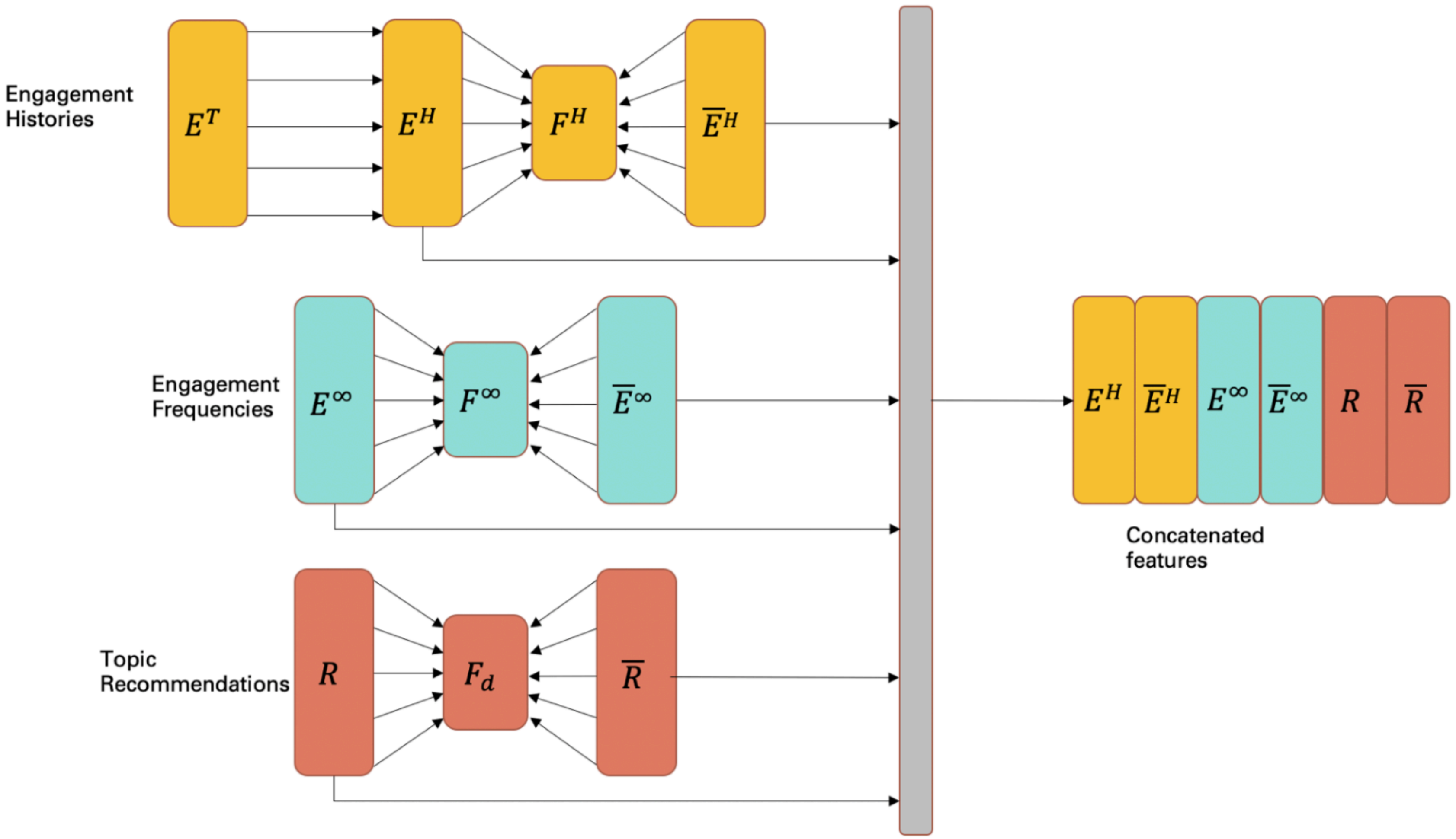
•

Text



Key Idea: If I show you these x items, which one(s) will you pick?
Has some overlap/difference with the idea of recommendations.

User Engagement Model based on Choices



User Engagement Model based on Choices

Given a collection of tweets shown to I users, we cluster them into J topics using LDA [Blei et al., 2003] such that each cluster contains tweets corresponding to a singular topical theme.

- Engagement vector: $\mathbf{e}_{it} = [e_{it0}, ..., e_{itj}] \in \{0,1\}^{J\times 1}$
- Engagement History:
  $\mathrm{E}_{it}^{T} = [\mathbf{e}_{i,t}, \mathbf{e}_{i,t-1}, ..., \mathbf{e}_{i,t-T+1}] \in \{0,1\}^{J\times T}$
- Engagement Frequency: $\mathrm{E}_{it}^{\infty} = [\bar{e}_{it0}, ..., \bar{e}_{itj}] \in [0,1]^{J\times 1}$
- Topic recommendation:
  $\mathrm{R}_{it} = [r_{it0}, .., r_{itj}] \in \{0,1\}^{J\times 1}$, where$r_{itj} \in \{0,1\}$
- $\mathrm{P}_{i,t+1} = f(R_{i,t+1}, E_{it}^{T}, E_{it}^{\infty}, \theta)$
- $\mathrm{p}_{i,t+1,j} = p(z_{i,t+1,j}; \theta_p)$, with
- $\mathbf{z}_{i,t+1} = [z_{i,t+1,1}, ..., z_{i,t+1,J}] \in \mathbb{R}^{J\times K}$and
  $\mathbf{z}_{i,t+1} = f^{'}(R_{i,t+1}, E_{it}^{T}, E_{it}^{\infty}; \theta_z)$.

User Engagement Model based on Choices

$$\theta = (\theta_z; \theta_p), \text{ and } \theta_z = (\mathbf{w}_{h=1...H}; W_d; W_\infty; W_H)$$

$$\theta^* = \arg\min_\theta \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{t=1}^{T} L(e_{i,t+1,j}, \hat{p}_{i,t+1,j})$$

User Engagement Model based on Choices

$$R_{it}^{*} \in \arg\max_{R=[r_1,\ldots,r_j]} \sum_{j} \hat{p}_{itj}(R, E_{it}^{T}, E_{it}^{\infty}).r_j$$

$$r_j \in \{0,1\}^{J\times 1} \text{and} \sum_{j} r_j = n$$

User Engagement Model based on Choices

- Data
  - Twitter (RecSys Challenge 2020)
  - 100k users, 3.4 million tweets
    - Userid
    - Tweetid
    - BERT tokens of the tweet text
    - Tweet/retweet timestamp
  - Convert data to account for choice based on timestamps

User Engagement Model based on Choices

	BCE loss	AUC score
Our Model	1.808	0.8601
Random Forest	2.145	0.5973
LightGBM	2.141	0.5320

BCE: Binary cross-entropy loss. AUC: Area under the ROC curve.

	Engagement Uplift Score
Our Model	0.365
Random Forest	0.209
LGBM	0.177

Uplift is computed by calculating the optimal solution using each model and scoring against the most expressive one. This can be calculated for each user and for each r time-window.

Questions?

Today's Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

Multi-armed Bandits under Priming Effect

With Priyank Agrawal (UIUC)

Paper: Learning by Repetition: Stochastic Multi-armed Bandits under Priming Effect

Conference: Uncertainty in Artificial Intelligence 2020

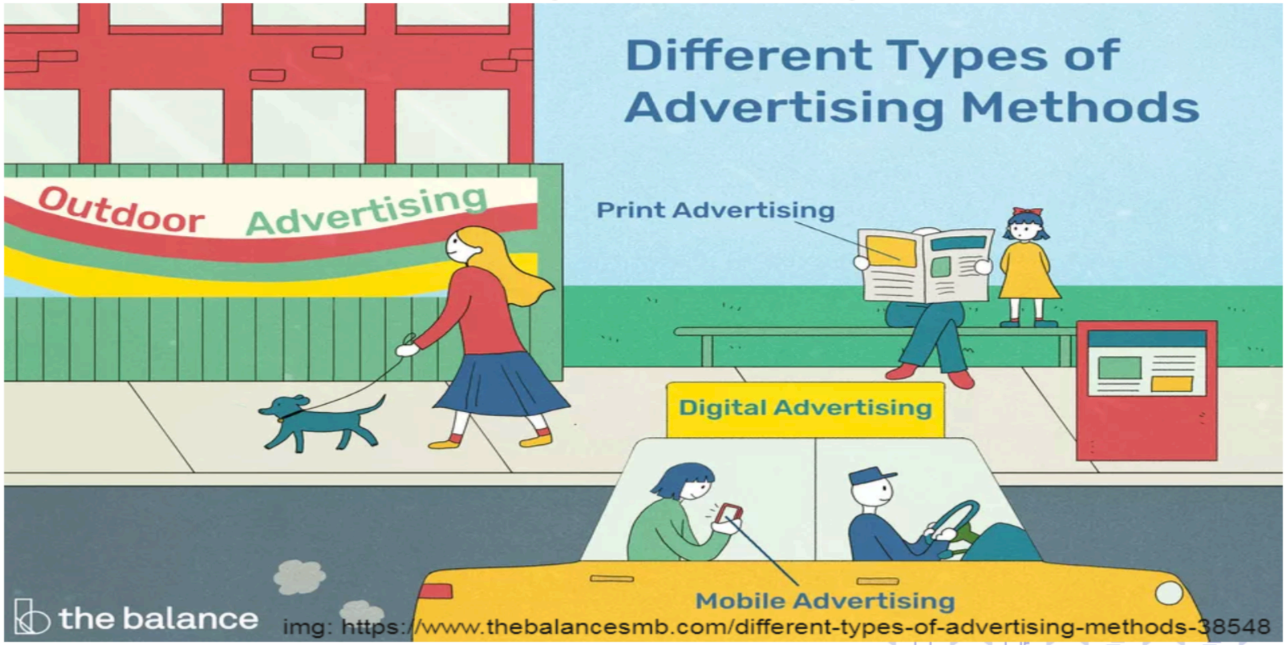
Multi-armed Bandits under Priming Effect

- Priming effect on consumer behavior:
  - Advertiser’s payoff depends on how frequently the consumer was presented with the same ad
  - Repeated display of recommendations can cause positive reinforcement.

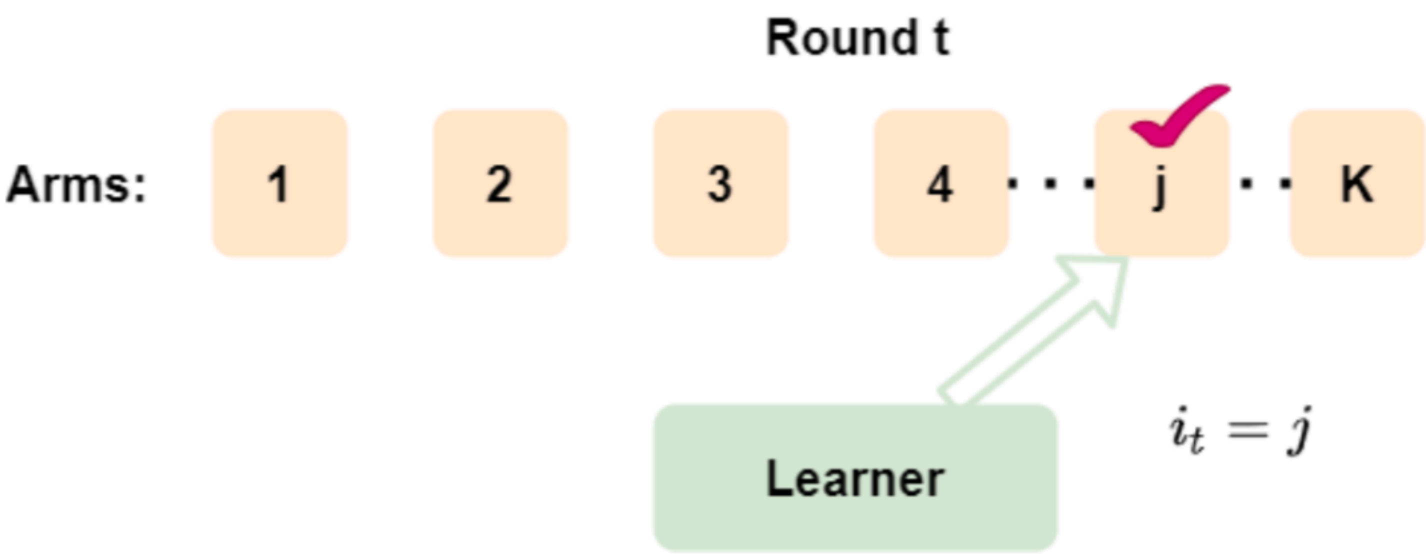
- Key contributions:
  - No need to use a full RL solution strategy
  - Use bandits where rewards depend on current and past actions
    - Advantage: get regret guarantees (upper bound on expected regret)

Multi-armed Bandits under Priming Effect

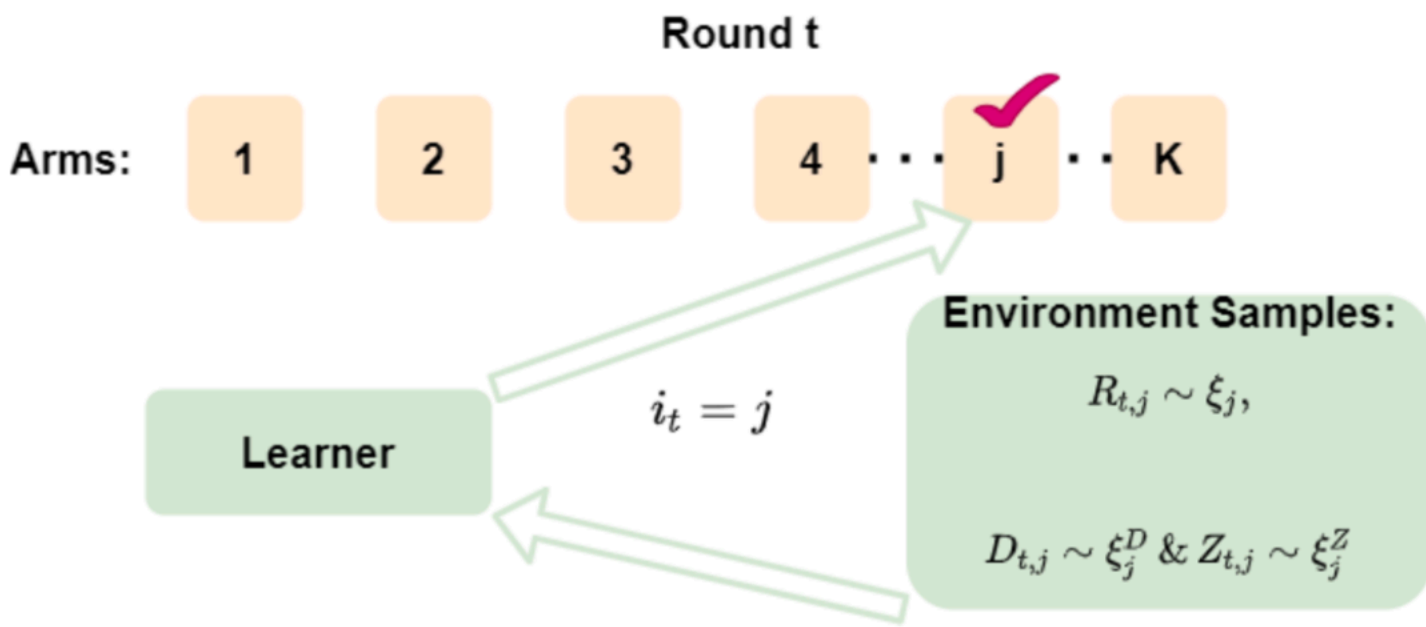
Effects of repetition: *wearing-in* **and** *wearing-out* **of the consumer.**



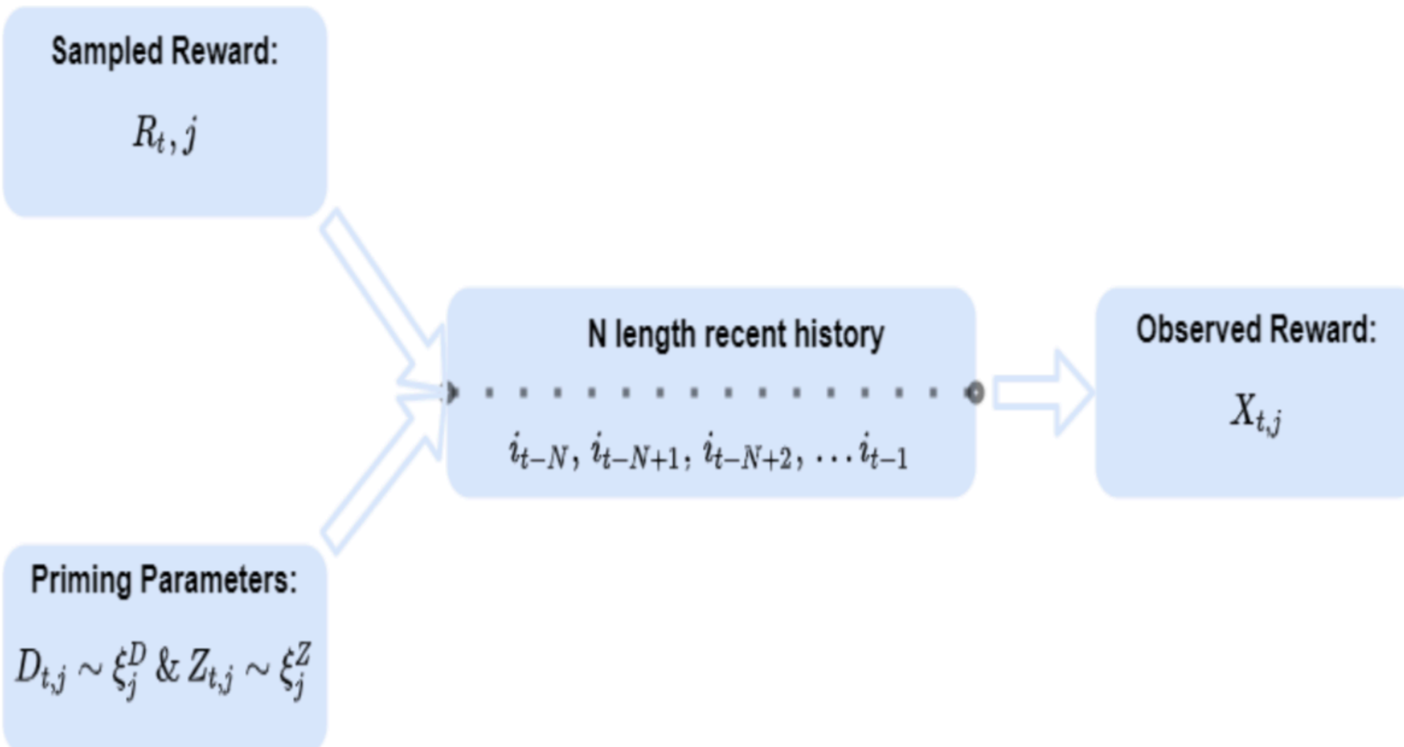
Multi-armed Bandits under Priming Effect



Multi-armed Bandits under Priming Effect



Multi-armed Bandits under Priming Effect



$f_{t,j}(N) \leftarrow$ History function.
The platform observes:
$X_{t,j} = R_{t,j}\mathbb{I}[f_{t,j}(N) \geq D_{t,j}]$ *(Only wear-in effect)*,
$X_{t,j} = R_{t,j}\mathbb{I}[Z_{t,j} \geq f_{t,j}(N) \geq D_{t,j}]$ *(wear-in & wear-out effects)*.

Multi-armed Bandits under Priming Effect

Multi-armed Bandits under Priming Effect

Algorithm details are omitted here.

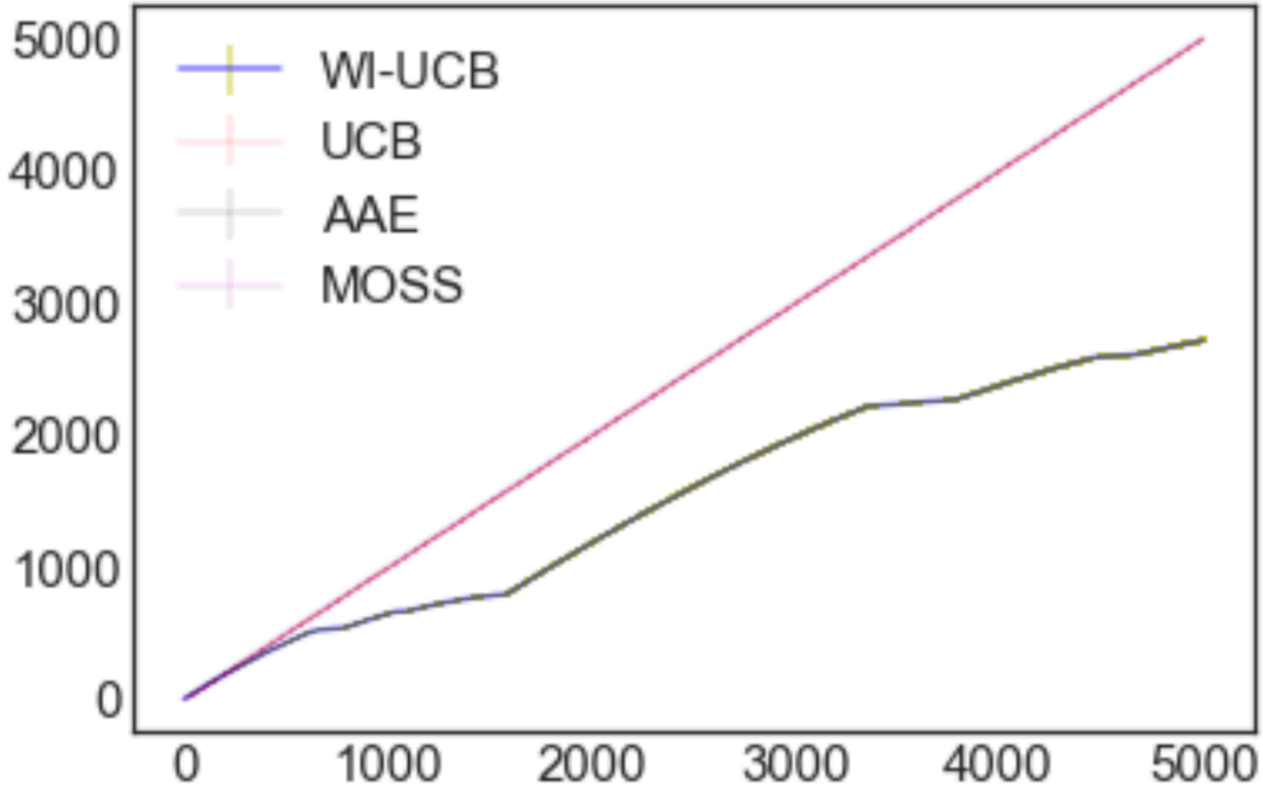


Figure 3: Performance (cumulative regret) of WI-UCB compared to other algorithms.

Multi-armed Bandits under Priming Effect

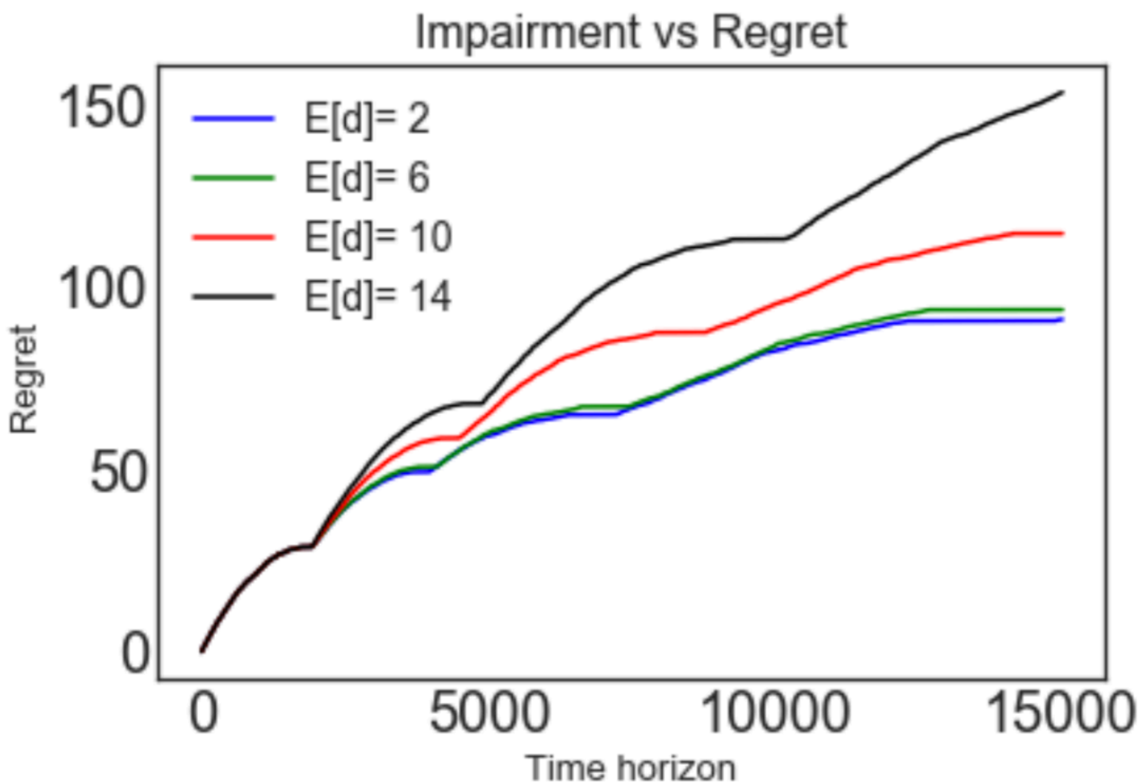


Figure 6: Performance (cumulative regret) of WI-UCB as the wear-in parameter is varied.

Questions?

Today's Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

Thompson Sampling for Recommendations

With Yunjuan Wang (UIC -> JHU)

Paper: Thompson Sampling for a Fatigue-aware Online Recommendation System

Venue: International Symposium on AI and Mathematics (2020)

Thompson Sampling for Recommendations

Platform Sends out a daily/weekly digest

User Clicks on interesting links *or* marks sender as spam

Platform Schedules a series of notifications for engagement

User Clicks on notifications and engages with app *or* mutes notifications forever

Thompson Sampling for Recommendations

- The platform recommends a (sub)-sequence $\mathbf{S}$ of items.
- User's intrinsic preference for item $j \in [N]$ is $u_j \in [0, 1]$.
- After viewing each item, the user can abandon the platform with probability $1 - q > 0$.
- If they abandon, the platform incurs a penalty $c > 0$.
- If they select j and leave, platform gets revenue $r_j > 0$.
- If they don't select j and move to the next item in $\mathbf{S}$, platform gets nothing.

Thompson Sampling for Recommendations

- Let $\mathbf{S} = (S_1, S_2, \ldots, S_m)$, where S_k denotes item in the k^{th} position
- Let $p_i(\mathbf{S})$ denote the probability of selecting item i in sequence $\mathbf{S}$.
- Let $p_a(\mathbf{S})$ denote the probability of total abandonment.

$$p_i(\mathbf{S}) = \begin{cases} u_i & \text{if } i \in S_1, \\ q^{l-1} \prod_{k=1}^{l-1} (1 - u_{S_k}) u_i & \text{if } i \in S_l, l \geq 2, \\ 0 & \text{if } i \notin \mathbf{S}. \end{cases}$$

$$p_a(\mathbf{S}) = \sum_{k=1}^{m} q^{k-1}(1-q) \prod_{j=1}^{k} \left(1 - u_{S_j}\right).$$

Thompson Sampling for Recommendations

The goal is to find the optimal sequence of items that maximizes expected utility $\mathbb{E}[U(\mathbf{S}; \mathbf{u}, q)] = \sum_{i \in \mathbf{S}} p_i(\mathbf{S}) r_i - c p_a(\mathbf{S})$:

$$\begin{aligned} \max_{\mathbf{S}} \quad & \mathbb{E}[U(\mathbf{S}; \mathbf{u}, q)] \\ \text{s.t.} \quad & S_i \cap S_j = \emptyset, \forall i \neq j, \\ & \text{and other business constraints,} \end{aligned}$$

where $\mathbb{E}[U(\mathbf{S}; \mathbf{u}, q)] = \sum_{i \in \mathbf{S}} p_i(\mathbf{S}) r_i - c p_a(\mathbf{S})$.

Thompson Sampling for Recommendations

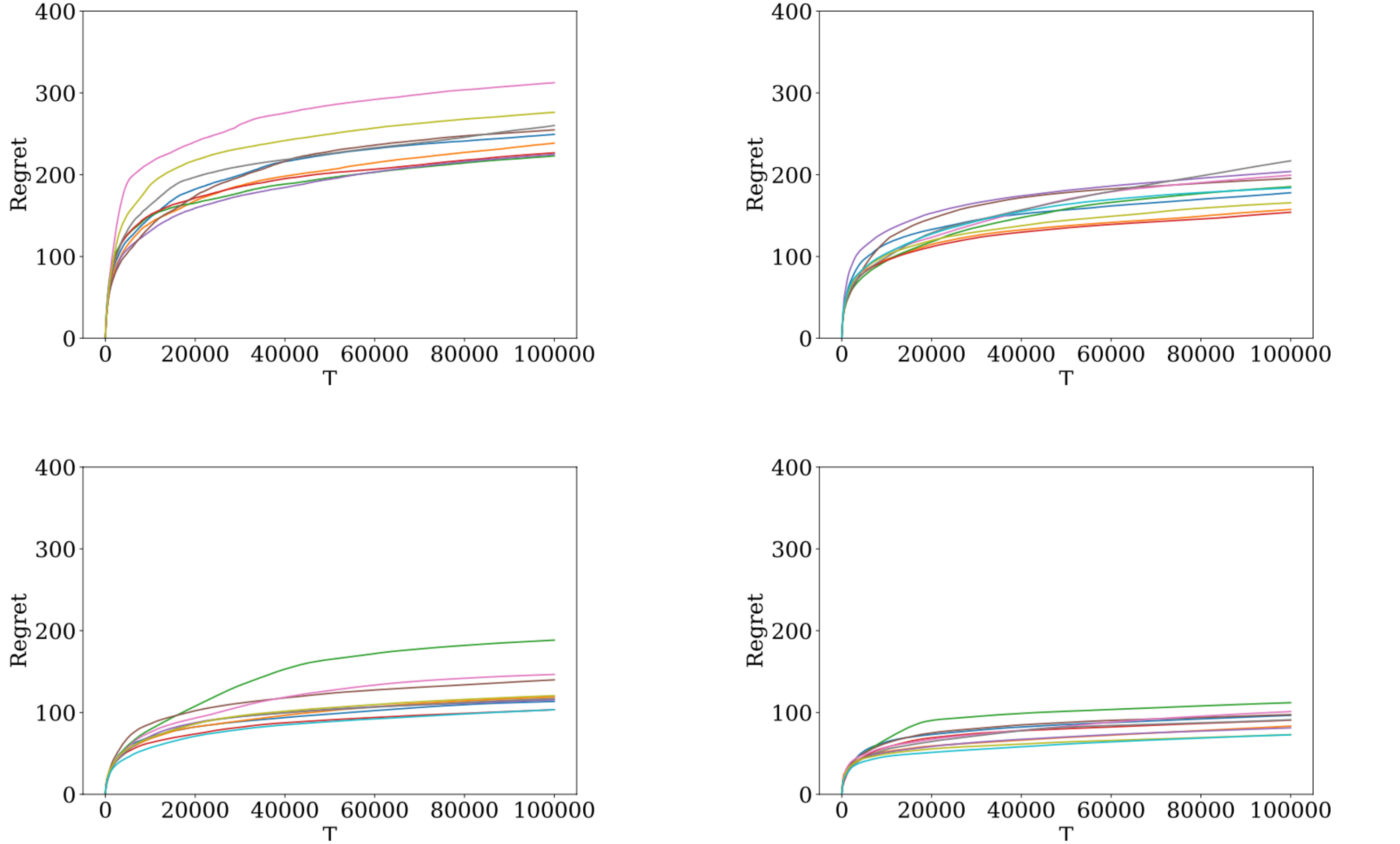
Algorithm details are omitted here.

A variation of the Thompson Sampling template.

In each round $1 \le t \le T$,

- Sample from the posterior belief over $(\mathbf{u}, q)$.
- Select the best sequence using the sample.
- Update the posterior depending on how the sequence fares.

Thompson Sampling for Recommendations



Here, N = 30, q = .9, c = .5 and $\mathbf{u}$ is uniformly generated from (a) [0,0.1], (b) [0,0.2], (c) [0,0.3], and (d) [0,0.5].

Questions?

Today's Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

RL for Optimization: Crowdsourced Last-Mile Urban Delivery

With Bo Zou, Tanvir Ahamed and Nahid Farazi (UIC)

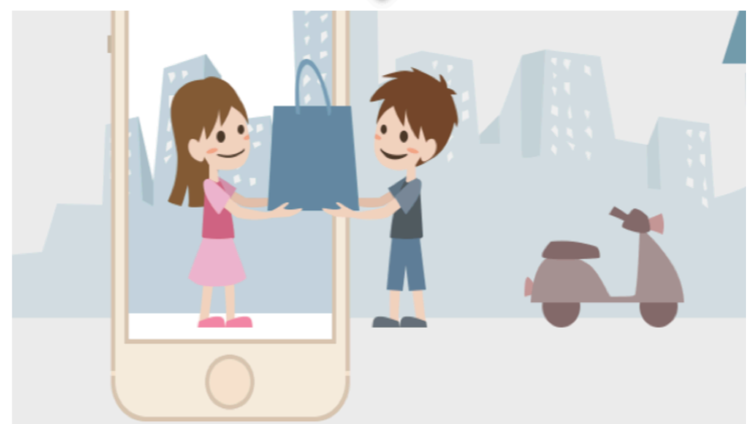
Paper: Rule-Interposing Deep Reinforcement Learning for Crowdsourced Last-Mile Urban Delivery

Draft Available (2020)

Presented at INFORMS 2020

RL for Optimization

Crowdshipping in this work concerns *intra-urban* shipments that can be picked up and delivered promptly using a crowd of *ordinary individuals* (**crowdsourcees**) who walk, bike, or drive to do delivery.



RL for Optimization

- An operation planning problem for crowdshipping
  - Consider a static case to perform request-crowdsourcee assignment
  - To be delivered in very short time, e.g., 1-2 hr.
  - If infeasible to assign, use backup vehicle
- Dedicated crowdsourcees
  - Limited available time
  - Limited carrying capacity
- Spatially distributed ODs of requests and starting locations of crowdsourcees

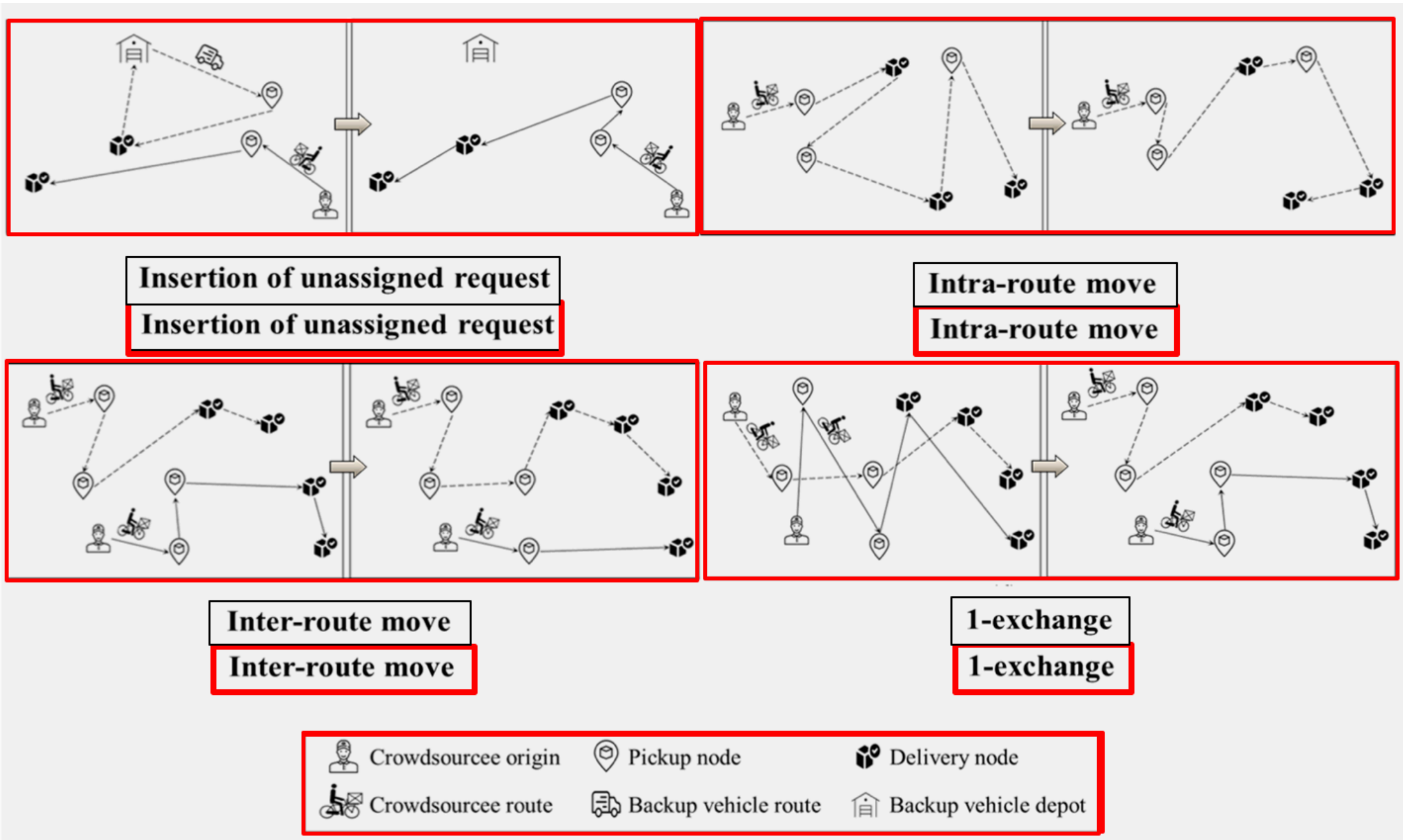
RL for Optimization: State

1. **Crowdsourcee specific information**
   - Location of the crowdsourcees
   - Remaining available time
   - Route duration
   - Extent of feasibility violations (time and capacity)

2. **Request specific information**
   - Slack time (how urgent a request)
   - Request pickup and delivery locations
   - Unused service time (gap between the latest delivery time and the actual delivery time)
   - Occupation time (duration between pickup and delivery of a request)

3. **Node adjacency information**
   - Node precedence relation of crowdsourcee routes

RL for Optimization: Actions

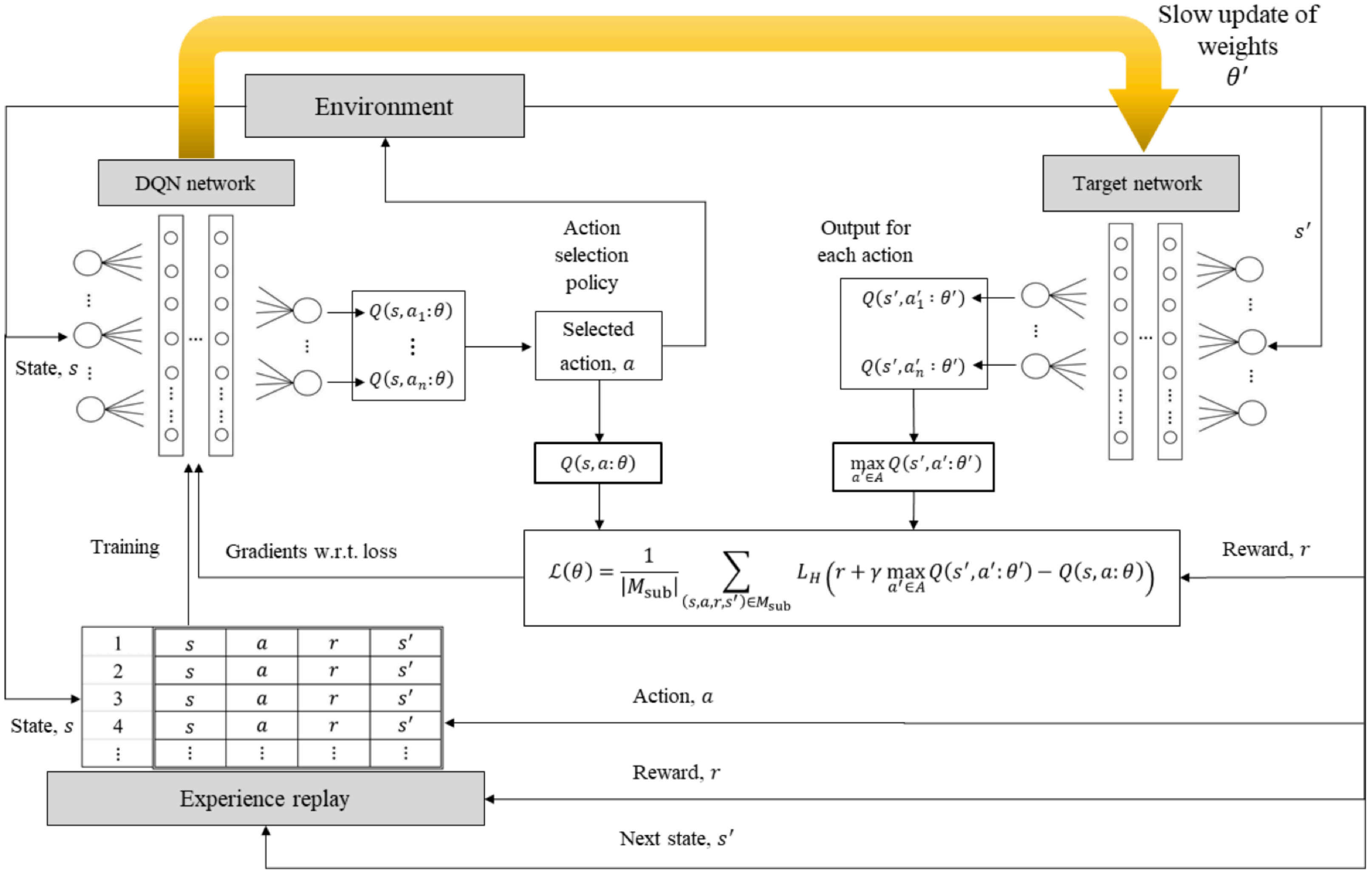


RL for Optimization: Reward

Reward

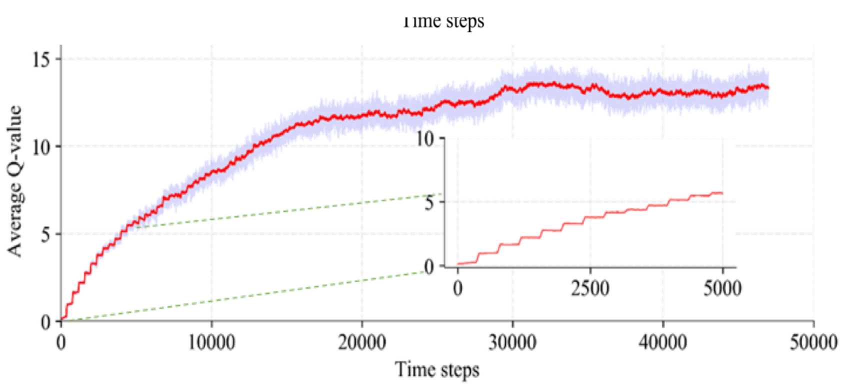
- The change in Total Shipping Cost (TSC) as a result of an action taken
- Penalty is imposed for feasibility violations

RL for Optimization: DQN Recap



RL for Optimization

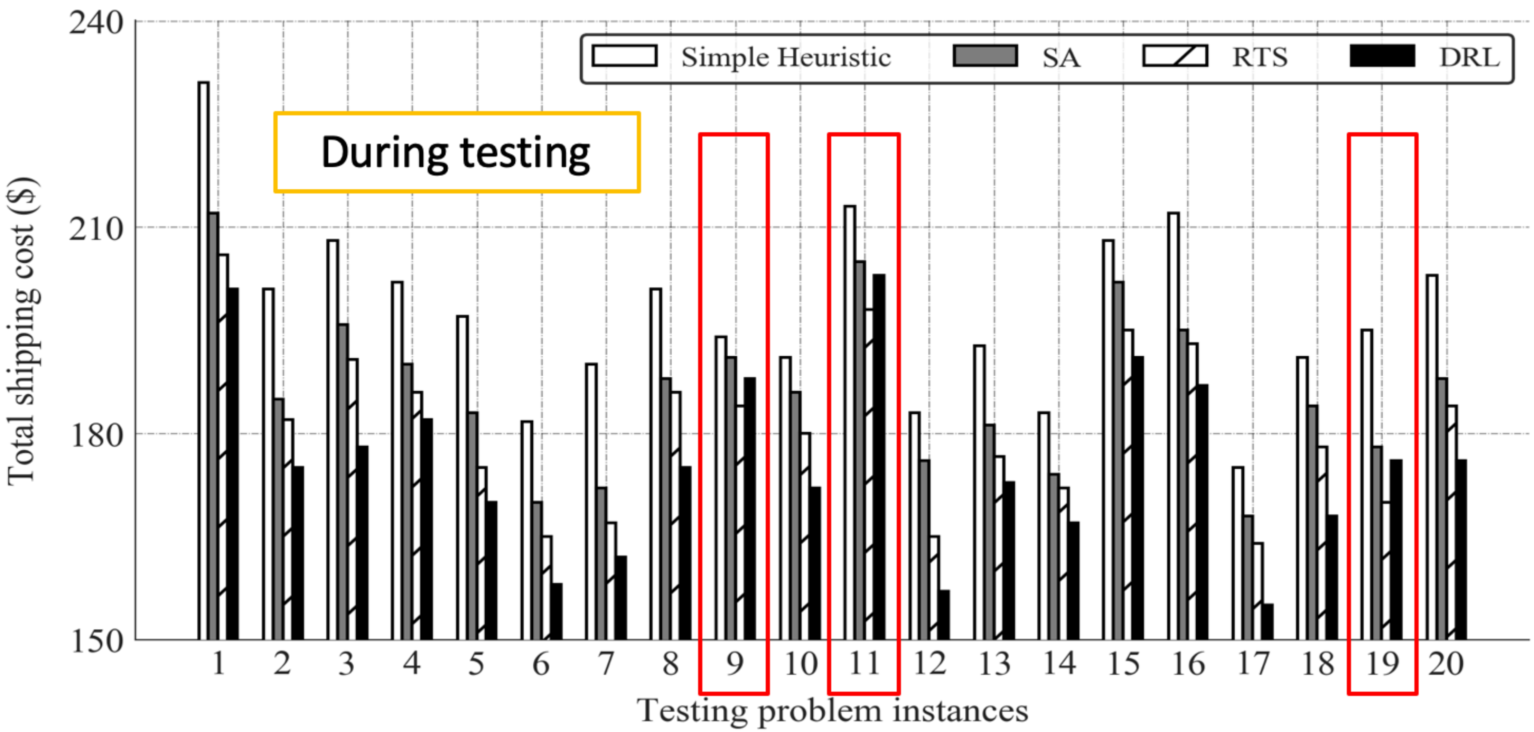
Training results



- Initially show increasing trend up to certain # of time steps
- Tends to stabilize afterwards
- Step-wise jumps in second Fig. correspond to target network update. diminishes gradually.
- Marginal improvement from training diminishes as training continues

Problem size: 50 requests and 22 crowdsourcees

RL for Optimization

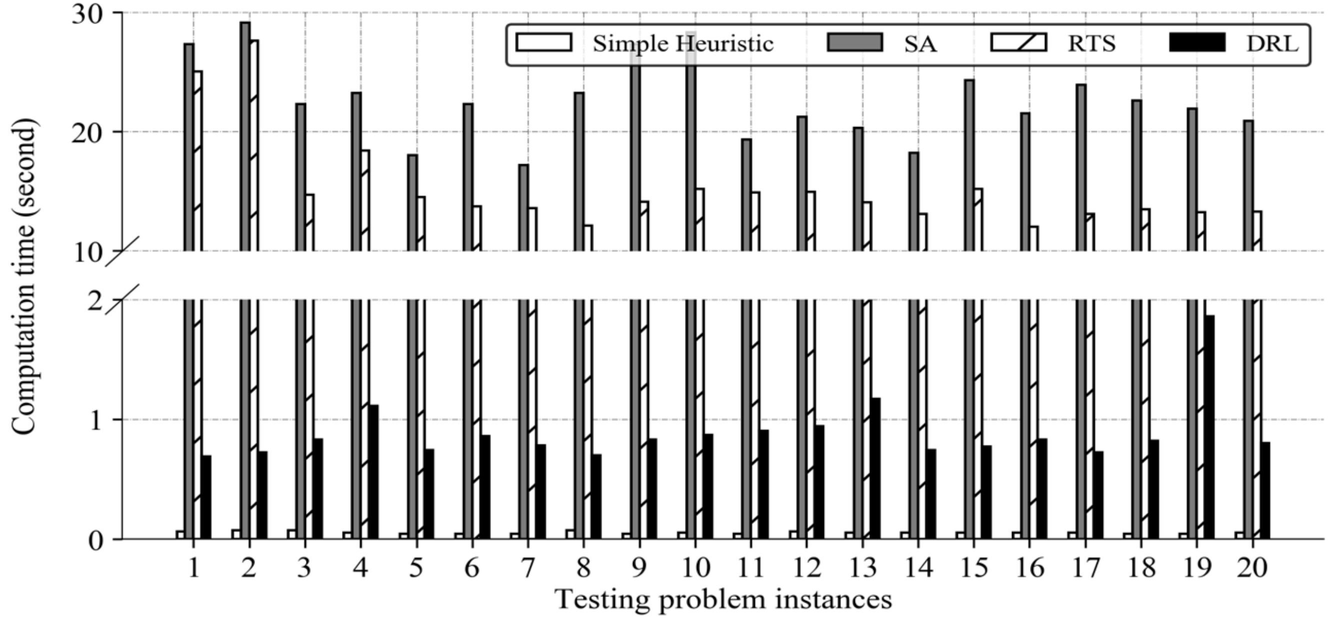


<u>Total shipping cost comparison</u>

DRL outperforms existing heuristics for **85%** of the problem instances (during testing).

RL for Optimization

During testing



Computation time comparison

Simple heuristic: <1 second
DRL: ~1 second
Tabu search: ~ 0.3 minute
Simulated annealing: ~ 0.5 minute

Questions?

Today’s Outline

- Knowledge Aware Attentive Sequential Recommendations (DL)
- User Engagement Model based on Choices (DL)
- Multi-armed Bandits under Priming Effect (Bandits)
- Thompson Sampling for Recommendations (Bandits)
- RL for Optimization: Crowdsourced Last-Mile Urban Delivery (RL)
- Improving RL by Detecting Symmetries (RL)

Improving RL by Detecting Symmetries

With Anuj Mahajan (Oxford)

Paper: Symmetry Detection and Exploitation for Function Approximation in Deep Reinforcement Learning

Venue: International Conference on Autonomous Agents and Multiagent Systems (2017)

Improving RL by Detecting Symmetries

- **RL algorithms are very slow** (e.g., Atari DQN)
- Humans are fast in finding good policies in various task settings
  - For various reasons ...
  - One reason could be because they exploit *symmetries*.
- Based on this motivation, we
  - develop a method to discover symmetries in an MDP.
  - Propose a way to exploit these (by modifying the DQN algorithm).
  - Validate this experimentally. Theory work still remains.

Improving RL by Detecting Symmetries

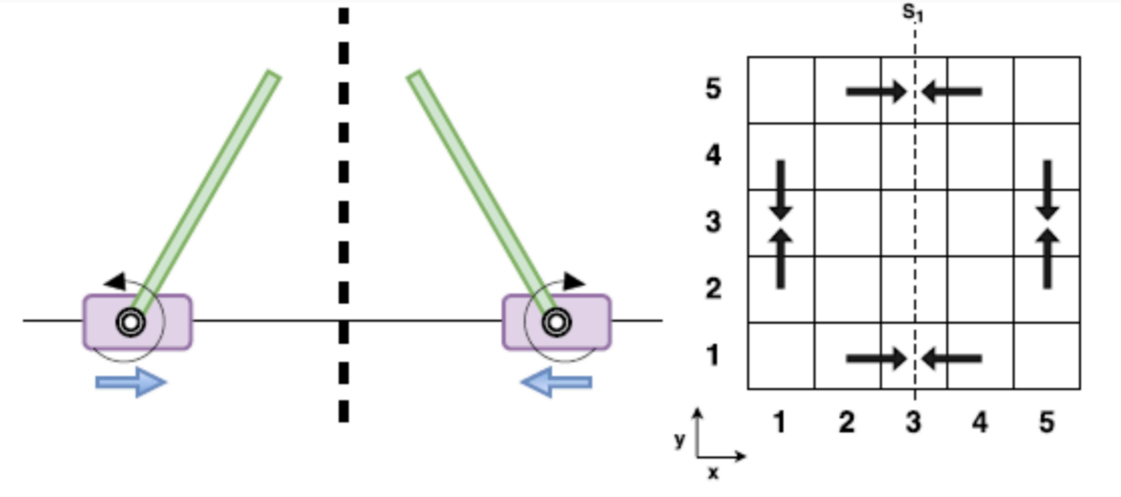


Figure 1: *Left*: Symmetry in Cart-Pole environment. *Right*: Symmetry in grid world. $f((x, y)) = (6 - x, y)$, $g_s : \{N \to N, S \to S, E \to W, W \to E\} \forall s$.

- Many real world settings exhibit symmetries.
- Symmetries tend to increase with the dimensionality of the state-action space. Eg. d dimensional grid world has $O(d!2^d)$ fold symmetries

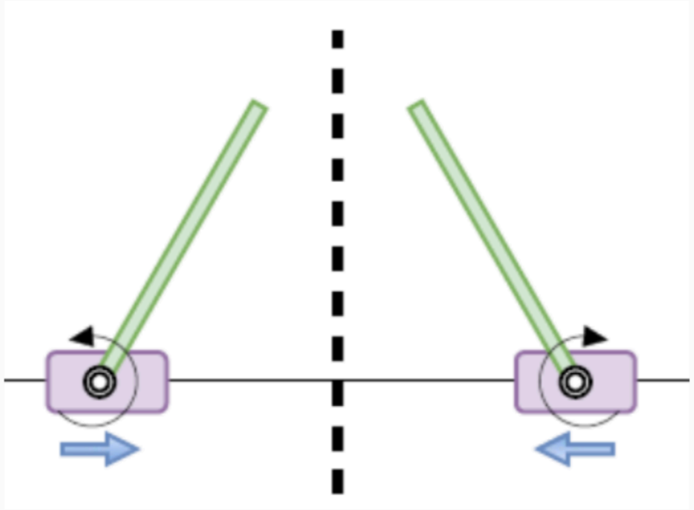
Improving RL by Detecting Symmetries

- Denote the VFA as $Q(s, a; \theta)$.
- Knowledge about symmetry is incorporated by adding a weighed symmetry based penalty to the usual TD training loss: $L_{i,total}(\theta_i) = L_{i,TD}(\theta_i) + \lambda L_{i,Sym}(\theta_i)$.

At iteration index i of our proposed algorithm **Sym DQN**, we have:

$$
\begin{aligned}
L_{i,TD}(\theta_i) &= \mathbf{E}_{\mathbf{B}}[((r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})) - Q(s, a; \theta_i))^2] \\
L_{i,Sym}(\theta_i) &= \mathbf{E}_{\chi_{sym}}[(Q(s', a'; \theta_i) - Q(s, a; \theta_i))^2]
\end{aligned}
$$

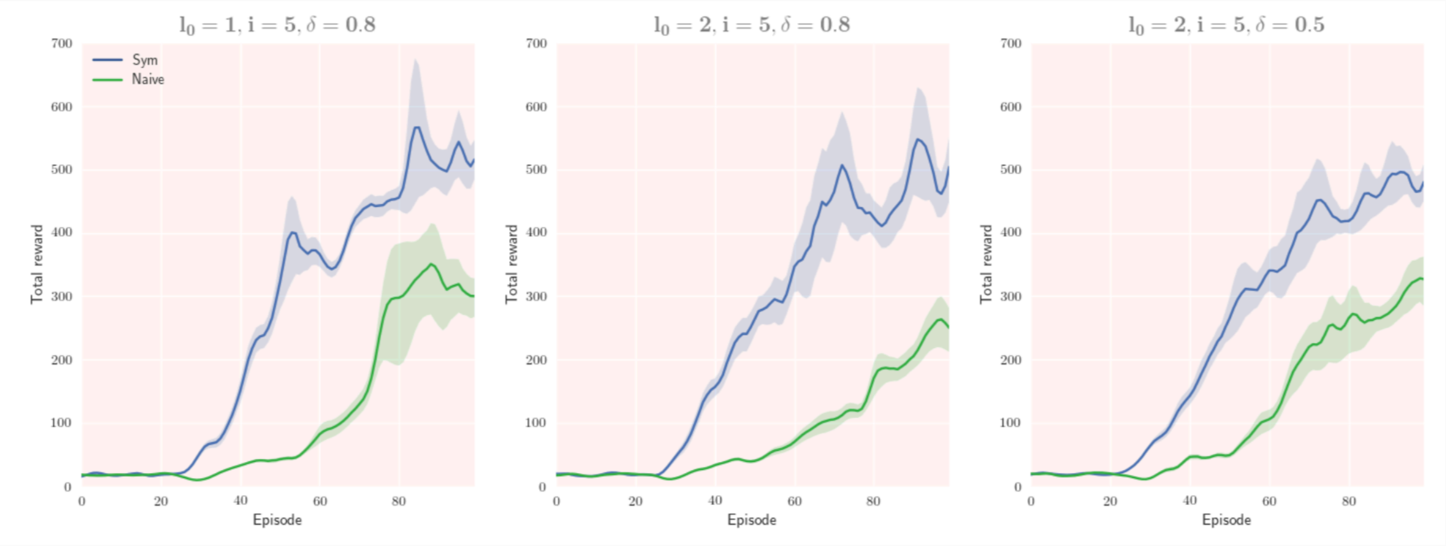
Improving RL by Detecting Symmetries



- Goal: Balance the pole on the cart for as long as possible.
  - States: (θ, x, ω, ν) bounded. Actions: Left/Right
  - Dynamics: physical model[2], Rewards: increasing towards target state
- Example symmetry: $((\theta, x, \omega, \nu)$,Left) and $((-\theta, -x, -\omega, -\nu)$,Right).

Improving RL by Detecting Symmetries

- Parameters $\lambda = 1, \gamma = 0.99$, and ϵ was decayed $1 \to 0.1$ at rate 0.98. Monte Carlo runs: 15.
- DQN/Sym DQN: replay memory size is 10^5, minibatch size is 128, 100 nodes each in 2 hidden layers.
- Using traditional rewards (+1 everytime pole is within bounds) is difficult here. Gives too many false positives.
- Reward plots show robustness to symmetry detection parameters.



Questions?

Thank You!