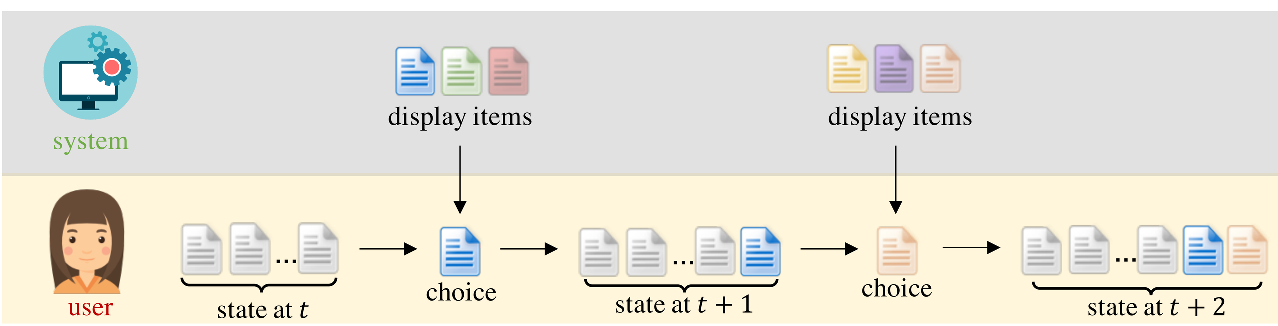
Generative Adversarial User Model for Reinforcement Learning Based Recommendation System

Xinshi Chen[1,†], Shuang Li[2], Hui Li[4], Shaohua Jiang[4], Yuan Qi[4], Le Song[3,4]

{[1]School of Mathematics, [2]Industrial and Systems Engineering, [3]College of Computing} at Georgia Institute of Technology, [4]Ant Financial Services Group

[†]Work done partially during an internship at Ant Financial.

Introduction: RL for Recommendation



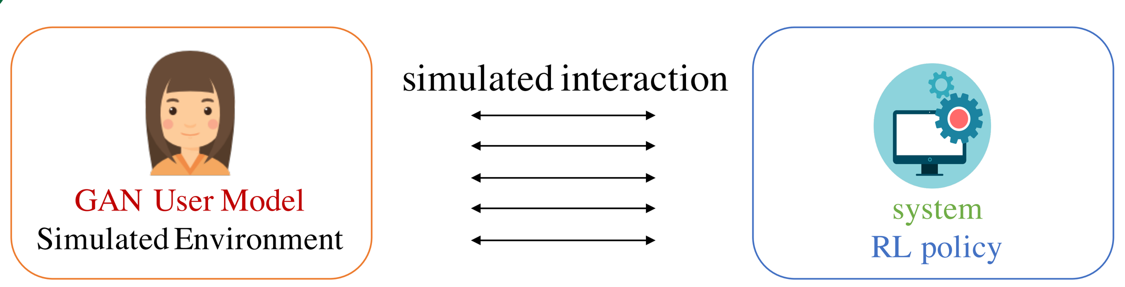
- A user's **interest evolves** over time based on what she observes.
- Recommender can significantly **influence such evolution**.
- **RL** based recommenders can consider user's **long term interest**.

Challenges.

1. User is the **environment** $\Longrightarrow$ Training of **RL** policy requires ***lots of interactions*** with users.
2. The **reward** function (a user's interest) is ***unknown***.

Our Solution and Contribution

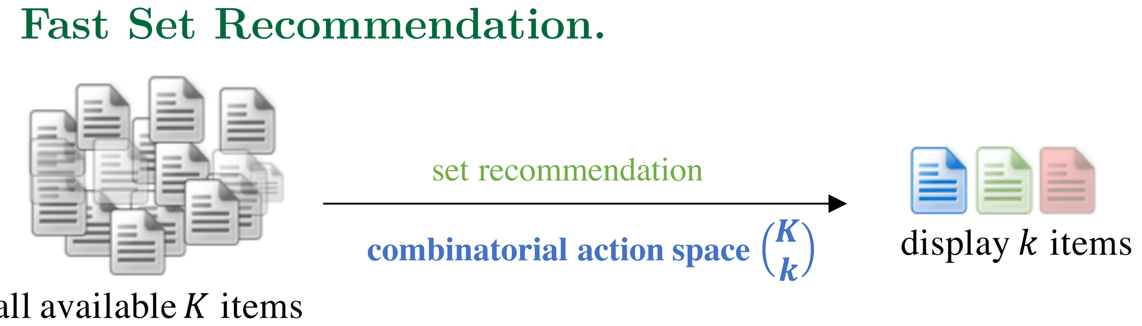
(1) GAN User Model as a Simulator.



We propose

- A **G**enerative **A**dversarial **User Model**
  - to model user's ***action***
  - to recover user's ***reward***
- Use GAN User Model as a **simulator**
  - to ***pre-train*** the **RL policy** offline.

(2) Fast Set Recommendation.



We design

- A **cascading Q** network to compute the optimal action in the ***combinatorial action space*** with only ***linear*** computation complexity.

Generative Adversarial User Model

The user model consist of 2 components:

1. User's *reward* $r(\boldsymbol{s}^t, a^t)$
   - a^t is the clicked item.
   - $\boldsymbol{s}^t$ is user's experience (state).
2. User's *strategy* $\phi^*(\boldsymbol{s}^t, \mathcal{A}^t)$
   - $\mathcal{A}^t$ contains items displayed by the system.
   - She will make a choice according to a strategy $a^t \sim \phi^*$ to maximize her expected reward.

Generative User Model:

$$\phi^*(\boldsymbol{s}^t, \mathcal{A}^t) = \arg\max_{\phi \in \Delta^{k-1}} \mathbb{E}_\phi\left[r(\boldsymbol{s}^t, a^t)\right] - R(\phi)/\eta$$

Model Parameterization.

Two architectures for aggregating historical information:

(1) LSTM.

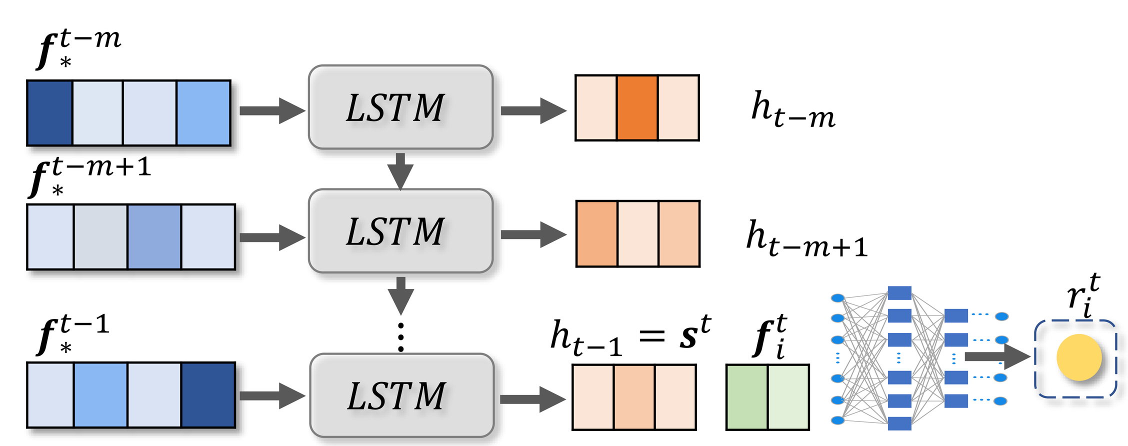


Figure: Architecture of user models parameterized by LSTM

(2) Position weight (PW).

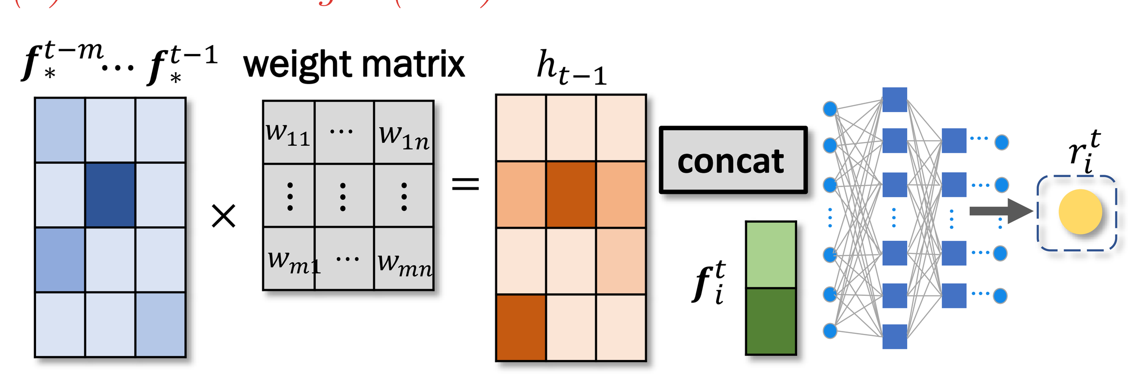


Figure: Architecture of user models parameterized by position weight (PW)

Generative Adversarial Training.

In analogy to generative adversarial networks (GAN):

1. *ϕ(strategy)* acts as a *generator* which generates user's next action based on her state.
2. *r(reward)* acts as a *discriminator* which tries to differentiate user's actual actions from those generated by the behavior model ϕ.

Mini-max Formulation:

$$\min_\theta \max_\alpha \left(\mathbb{E}_{\phi_\alpha}\left[\sum_{t=1}^T r_\theta(\boldsymbol{s}^t_{true}, a^t)\right] - R(\phi_\alpha)/\eta\right) - \sum_{t=1}^T r_\theta(\boldsymbol{s}^t_{true} a^t_{true})$$

Set Recommendation RL policy: Cascading DQN

The challenge is that the recommendation policy needs to choose from a **combinatorial action space** $\binom{\mathcal{I}}{k}$.

$$a_1^*, a_2^*, \ldots, a_k^* = \arg\max_{a_1, \ldots, a_k} Q(\boldsymbol{s}^t, a_1, a_2, \ldots, a_k)$$

Intractable Computation!

We design a **cascading Q** network to compute the optimal action in $O(k|\mathcal{I}|)$ computations:

Cascading Q-Networks:

$$a_1^* = \arg\max_{a_1}\{\mathbf{Q}^{1*}(\boldsymbol{s}, a_1) := \max_{a_{2:k}} Q(\boldsymbol{s}, a_{1:k})\},$$
$$a_2^* = \arg\max_{a_2}\{\mathbf{Q}^{2*}(\boldsymbol{s}, a_1^*, a_2) := \max_{a_{3:k}} Q(\boldsymbol{s}, a_{1:k})\},$$
$$\cdots$$
$$a_k^* = \arg\max_{a_k}\{\mathbf{Q}^{k*}(\boldsymbol{s}, a_{1:k-1}^*, a_k) := Q(\boldsymbol{s}, a_{1:k})\}.$$

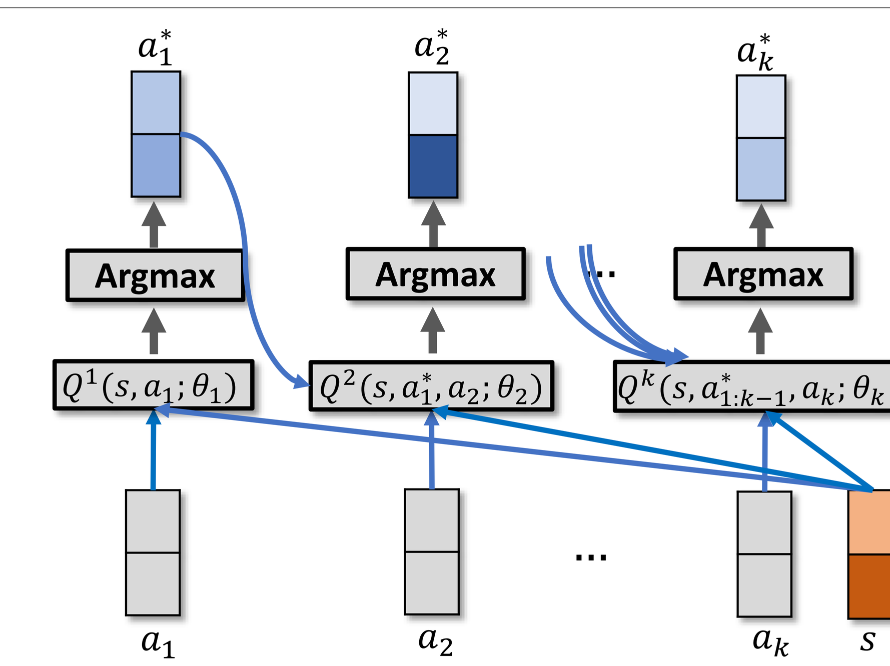


Figure: Cascading Q-networks.

Estimation of Q Functions.

The set of Q^{j*} functions need to satisfy

$$Q^{j*}(\boldsymbol{s}, a_1^*, \cdots, a_j^*) = Q(\boldsymbol{s}, a_1^*, \cdots, a_k^*), \quad \forall j.$$

We take them into account in a soft and approximate way by defining the loss as

$$\left(y - Q^j\right)^2, \text{ where } y = r(\boldsymbol{s}^t, \mathcal{A}^t, a^t) + \gamma Q^k(\boldsymbol{s}^{t+1}, a^*_{1:k}; \Theta_k), \ \forall j.$$

All Q^j networks are fitting against the same target y. In our experiments the set of learned Q^j networks satisfies the constraints nicely with a small error:

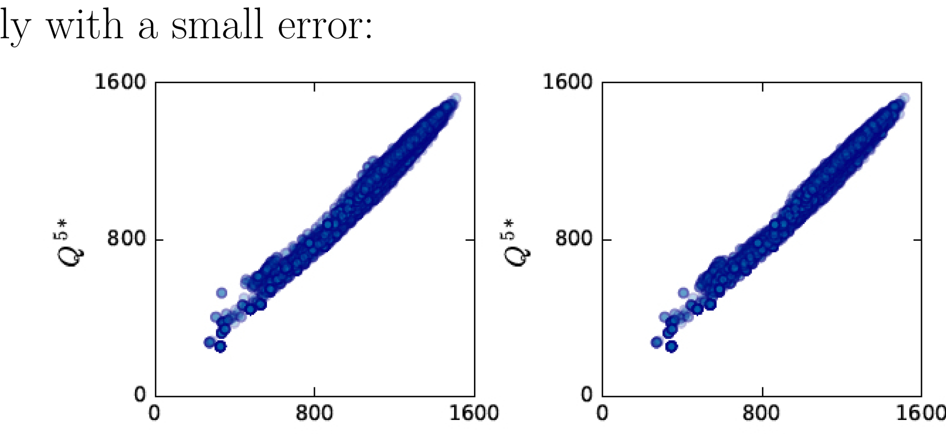


Figure: Each scatter-plot compares Q^{j*} with Q^{5*} evaluated at the same set of k recommended items. In ideal cases, all points should lie along the diagonal.

Experiments

Predictive Performance of User Models.

	(1) MovieLens		(2) LastFM		(6) Ant Financial	
Model	prec(%)@1	prec(%)@2	prec(%)@1	prec(%)@2	prec(%)@1	prec(%)@2
IKNN	38.8(±1.9)	40.3(±1.9)	20.4(±0.6)	32.5(±1.4)	20.6(±0.2)	32.1(±0.2)
S-RNN	39.3(±2.7)	42.9(±3.6)	9.4(±1.6)	17.4(±0.9)	32.2(±0.9)	40.3(±0.6)
SCKNNC	49.4(±1.9)	51.8(±2.3)	21.4(±0.5)	26.1(±1.0)	34.6(±0.7)	43.2(±0.8)
XGBOOST	66.7(±1.1)	76.0(±0.9)	10.2(±2.6)	19.2(±3.1)	41.9(±0.1)	65.4(±0.2)
DFM	63.3(±0.4)	75.9(±0.3)	10.5(±0.4)	20.4(±0.1)	41.7(±0.1)	64.2(±0.2)
W&D-LR	61.5(±0.7)	73.8(±1.2)	7.6(±2.9)	16.6(±3.3)	37.5(±0.2)	60.9(±0.1)
W&D-CCF	65.7(±0.8)	75.2(±1.1)	15.4(±2.4)	25.7(±2.6)	37.7(±0.1)	61.1(±0.1)
GAN-PW	66.6(±0.7)	75.4(±1.3)	**24.1**(±0.8)	**34.9**(±0.7)	41.9(±0.1)	65.8(±0.1)
GAN-LSTM	**67.4**(±0.5)	**76.3**(±1.2)	24.0(±0.9)	34.9(±0.8)	**42.1**(±0.2)	**65.9**(±0.2)

Recommendation Policy Based on User Model.

	$k=3$		$k=5$	
model	reward	ctr	reward	ctr
W&D-LR	14.46(±0.42)	0.46(±0.01)	15.18(±0.38)	0.48(±0.01)
W&D-CCF	19.93(±1.09)	0.62(±0.03)	20.94(±1.03)	0.65(±0.03)
GAN-Greedy	21.37(±1.24)	0.67(±0.04)	22.97(±1.22)	0.71(±0.03)
GAN-RWD1	22.17(±1.07)	0.68(±0.03)	25.15(±1.04)	**0.78**(±0.03)
GAN-GDQN	23.60(±1.06)	0.72(±0.03)	23.19(±1.17)	0.70(±0.03)
GAN-CDQN	**24.05**(±0.98)	**0.74**(±0.03)	**25.36**(±1.10)	0.77(±0.03)
DQN-Off	20.31(±0.14)	0.63(±0.01)	21.82(±0.08)	0.67(±0.01)

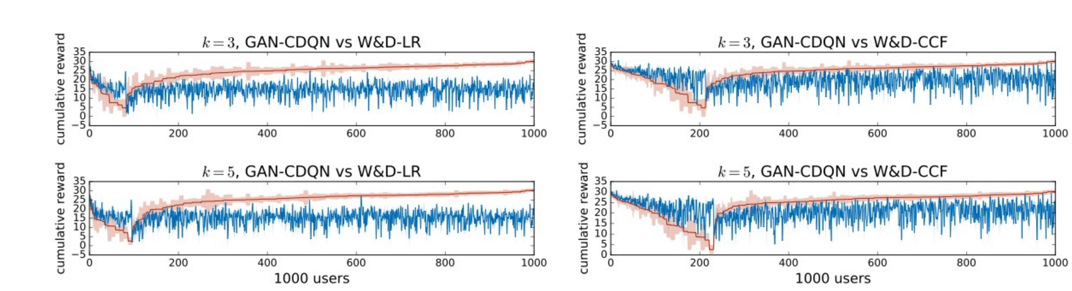


Figure: Cumulative rewards among 1,000 users under the recommendation policies based on different user models.

User Model Assisted Policy Adaptation.

Cascading-DQN policy pre-trained over a GAN user model can quickly achieve a high CTR even when it is applied to a new set of users.

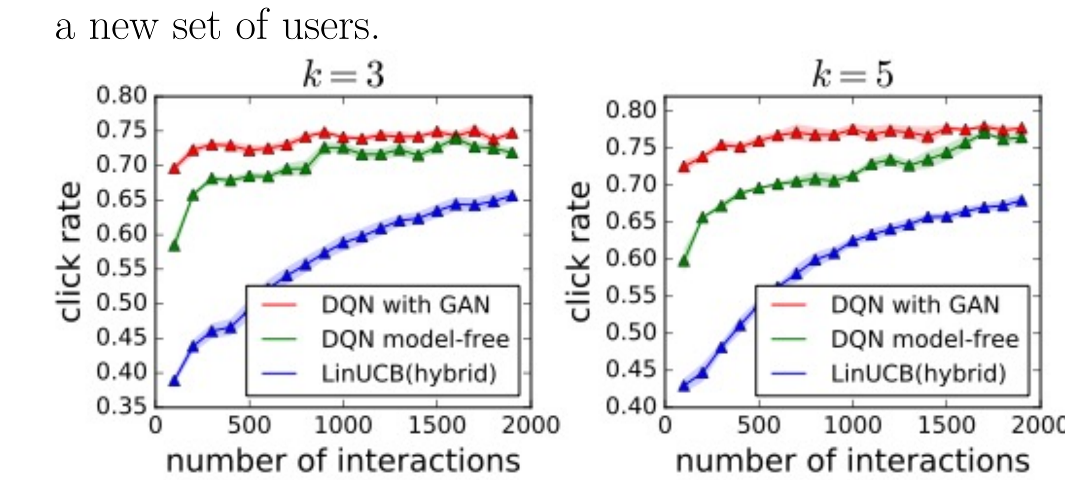


Figure: Comparison of the averaged click rate averaged over 1,000 users under different recommendation policies. X-axis represents how many times the recommender interacts with online users. Y-axis is the click rate. Each point (x, y) means the click rate y is achieved after x times of user interactions.

Contact

Xinshi Chen: xinshi.chen@gatech.edu

Le Song: lsong@cc.gatech.edu