Deep Reinforcement Learning for Page-wise Recommendations

11/12/2021

Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang

Presented by: Arash Moayyedi

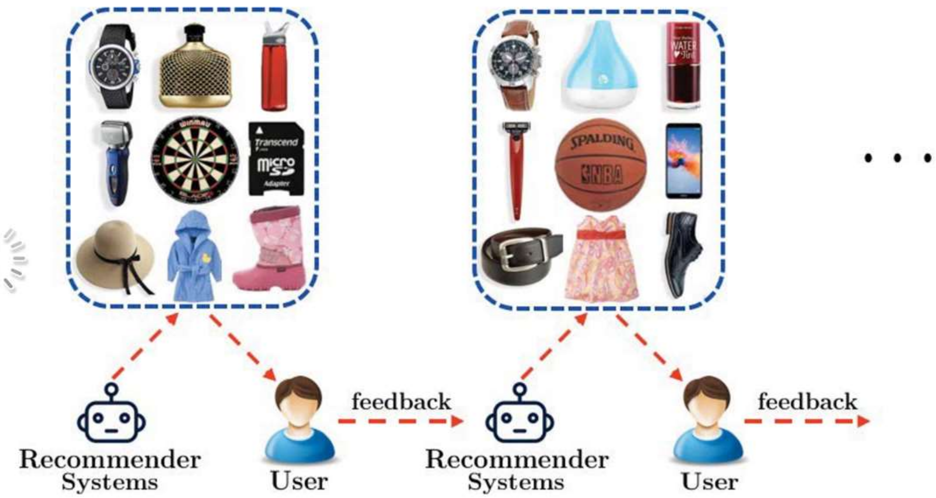
UNIVERSITY OF WATERLOO

Recommendation Systems

Personalized suggestions

Two challenges:

- Updating suggestions based on feedback
- Properly displaying the suggestions



Previous Works

- Mostly greedy static processes
- Fail to continuously update their strategies
- Similar repeating suggestion sets
- Do no consider item placement in page

Our Work

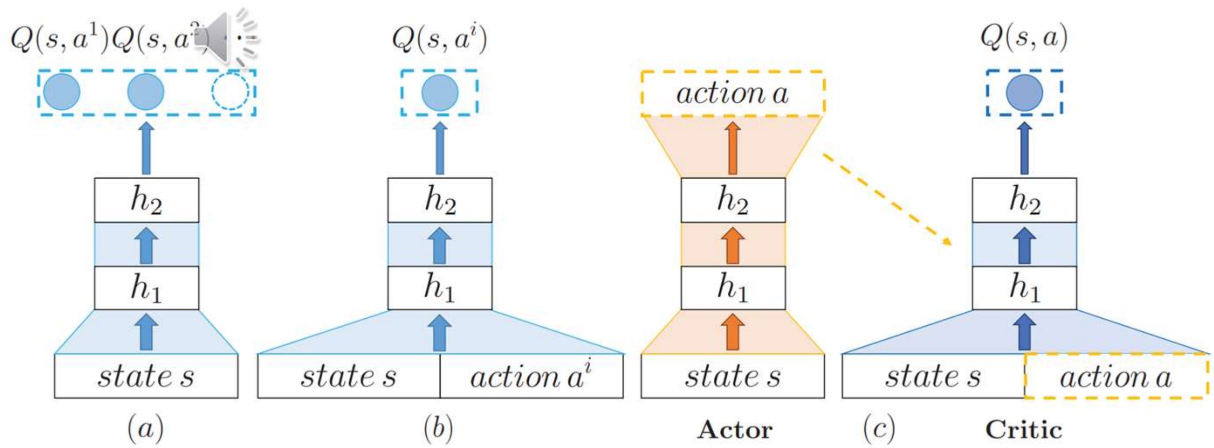
- Leveraging Deep Reinforcement Learning
- Capturing and utilizing users' real-time feedbacks
- Maximizing long-term rewards
- Bundling diverse and complimentary items
- Forming item display strategies for 2D pages

Choice of Framework Architecture

a) Conventional DQN methods do not support large action spaces

b) The time-complexity makes it impractical

c) Suitable for large and dynamic action spaces



UNIVERSITY OF WATERLOO

Architecture of Actor

Three challenges

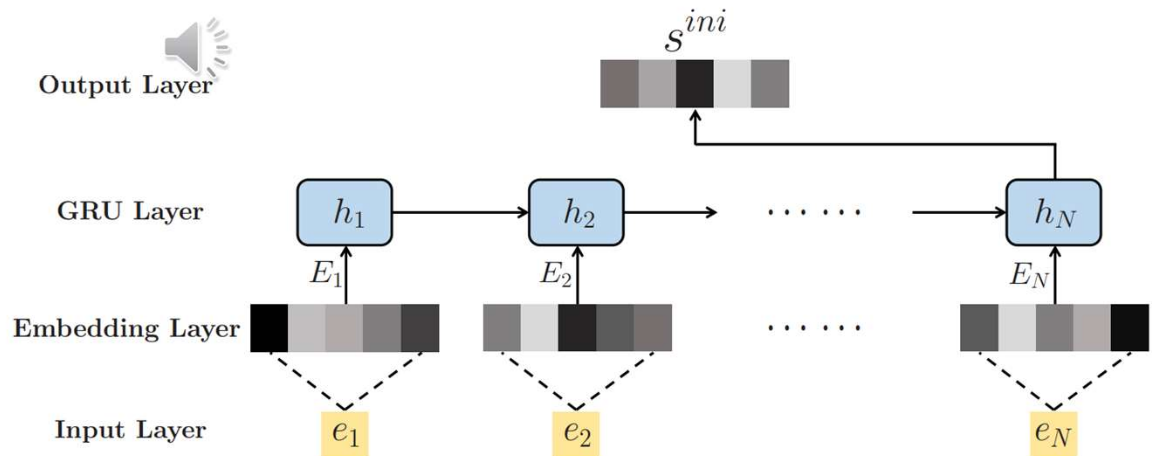
1. Setting an initial preference at the beginning of the session
2. Learning the real-time preferences in the current session
3. Jointly generating a set of recommendations and arranging them

Initial State Generation

Input Layer: user's last clicked/purchased items' representations

Embedding Layer: $E_i = \tanh(W_E e_i + b_E) \in R^{|E|}$

GRU: Gated Recurrent Units to capture users' sequential behavior



Real-time State Generation

Input Layer: x_i = (e_i = item rep., c_i = item cat., f_i = user feedback)

Embedding Layer:

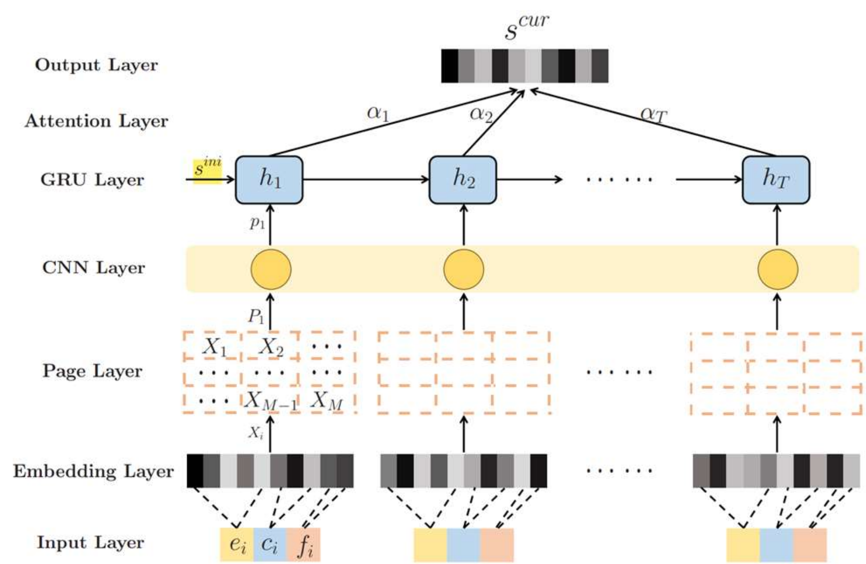
- $E_i = \tanh(W_E e_i + b_E) \in R^{|E|}$
- $C_i = \tanh(W_C c_i + b_C) \in R^{|C|}$
- $F_i = \tanh(W_F f_i + b_f) \in R^{|F|}$

Page Layer: original arrangement

CNN Layer: capture spatial correlations

GRU Layer: capture temporal correlations

Attention Layer: Weighted inputs



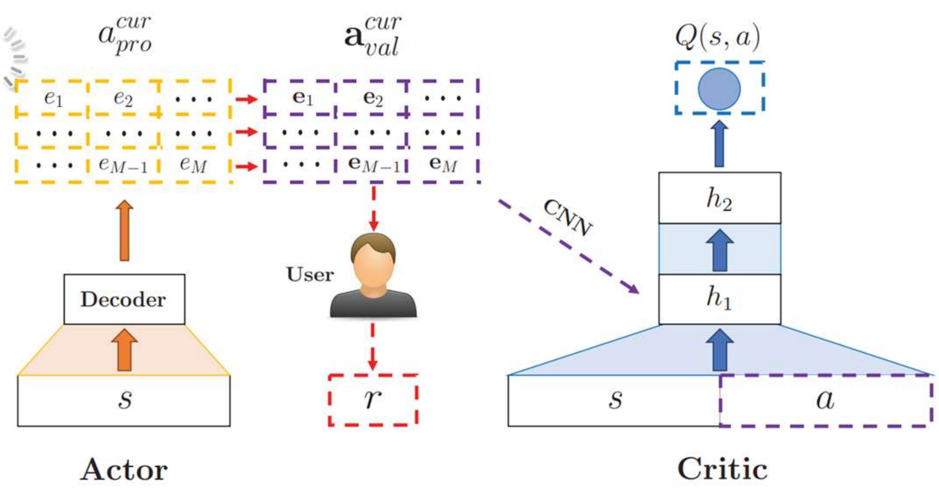
UNIVERSITY OF WATERLOO

Action Generation

- Converts the vectors to 2D item lists
- Uses deconvolution neural networks (DeCNN)
- Output might need to be matched to actual items

Architecture of Critic

- Uses 2D-CNN to extract action vector from item page
- Action vector and state vector input to DQN



UNIVERSITY OF WATERLOO

Training

DDPG

Offline

- Pretrain agent with offline data
- Off-policy
- Fixed action data
- Minimize actor output and data difference:

$$\min_{\theta^\pi} \sum_{b=1}^{B} \left(\| a_{pro}^{cur} - \mathbf{a}_{val}^{cur} \|_F^2 \right)$$

Online

- On-policy
- Output matched through cosine similarity:

$$\mathbf{e}_i = \arg\max_{\mathbf{e} \in \mathcal{I}} \frac{e_i^\top \cdot \mathbf{e}}{\|e_i\| \|\mathbf{e}\|} = \arg\max_{\mathbf{e} \in \mathcal{I}} e_i^\top \cdot \frac{\mathbf{e}}{\|\mathbf{e}\|}$$

- Rewards based on actual user interactions

Training Algorithm

Critic

- Minimize loss function: $L(\theta^{\mu}) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma Q_{\theta^{\mu'}}(s', f_{\theta^{\pi'}}(s')) - Q_{\theta^{\mu}}(s, a)\right)^2\right]$
- Trained from samples in replay buffer

Actor

- Policy gradient update: $\nabla_{\theta^{\pi}} f_{\theta^{\pi}} \approx \mathbb{E}_s\left[\nabla_{\hat{a}} Q_{\theta^{\mu}}(s, \hat{a}) \nabla_{\theta^{\pi}} f_{\theta^{\pi}}(s)\right]$

Testing Procedure

Online testing

- Similar to transition generating stage during training

Offline testing

- Reranking items

Experiments

Database: 1M recommendation sessions

Initial state: 10 previous items

Recommendation page: 5 rows, 2 columns

Reward:

- Skipped: 0
- Clicked: 1
- Purchased 5

Embedding dimensions:

- $|E| = 50$
- $|C| = 35$
- $|F| = 15$

Offline metrics:

- Precision@20,
- Recall@20,
- F1-score@20,
- NDCG@20
- MAP

Online metrics:

- total sum of rewards

Results (Offline Test)

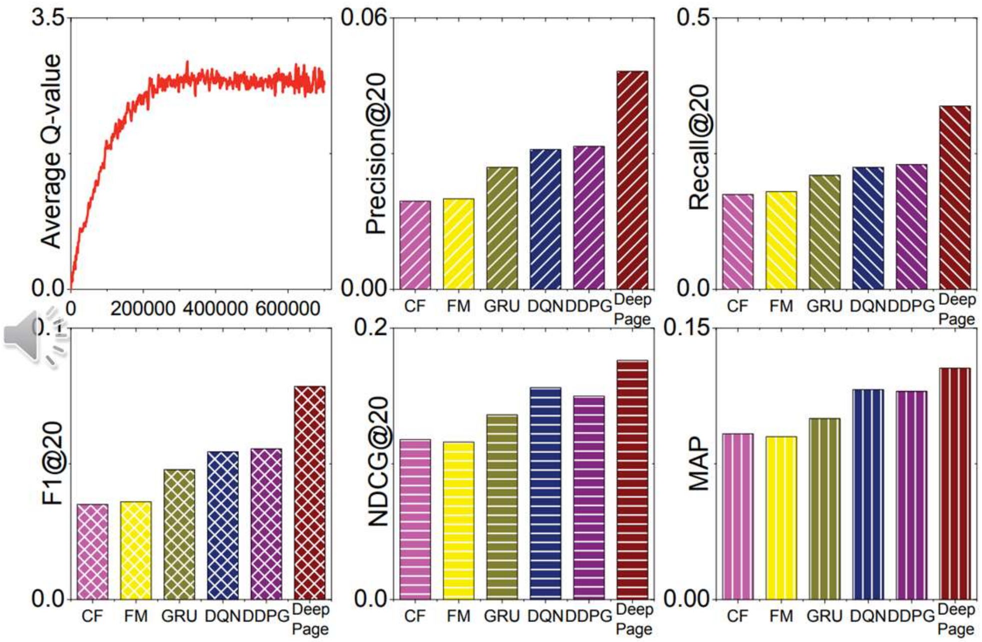
CF: Collaborative Filtering

FM: Factorization Machines

GRU: Click/purchase prediction

DQN: User history -> Recom. page

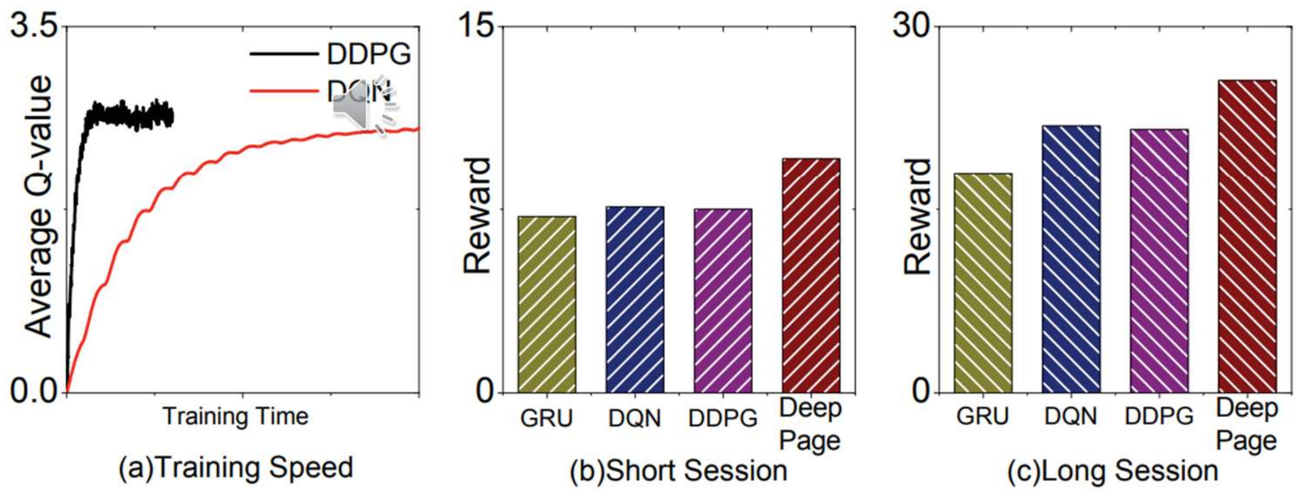
DDPG: Using 5 fully connected layers



Results (Online Test)

Short session: 10 recommendation pages

Long session: 50 recommendation pages



UNIVERSITY OF WATERLOO

Effectiveness of Components

1: remove embedding layers

2: remove categories and feedback

3: remove GRU in initial state

4: remove CNN

5: remove attention mechanisms

6: remove GRU in real-time state

7: replace DeCNN with FCL

Methods	Precision @20	Recall @20	F1score @20	NDCG @20	MAP
DeepPage-1	0.0479	0.3351	0.0779	0.1753	0.1276
DeepPage-2	0.0475	0.3308	0.0772	0.1737	0.1265
DeepPage-3	0.0351	0.2627	0.0578	0.1393	0.1071
DeepPage-4	0.0452	0.3136	0.0729	0.1679	0.1216
DeepPage-5	0.0476	0.3342	0.0775	0.1716	0.1243
DeepPage-6	0.0318	0.2433	0.0528	0.1316	0.1039
DeepPage-7	0.0459	0.3179	0.0736	0.1698	0.1233
DeepPage	**0.0491**	**0.3576**	**0.0805**	**0.1872**	**0.1378**

Conclusion

- Recommendation systems are complicated tasks
- Deep Reinforcement Learning proves necessary

Future work

- Reducing the temporal complexity of mapping output embeddings to actual items
- Handling multiple tasks in one reinforcement learning framework


That's all Folks!