**Motivation**

Recommender systems can mitigate the information overload problem by suggesting users’ personalized items

- **Challenges of Existing Recommender Systems**
  - Recommendation procedure as static process
  - Making recommendations following fixed greedy strategy
  - Maximizing the immediate (short-term) reward from users

- **Why Reinforcement Learning?**
  - Continuously updating the recommendation strategies
  - Maximizing the long-term reward from users

- **Why Negative Feedback?**
  - Positive: click or purchase
  - Negative: skip
  - What users may not like

**Goal:** find a recommendation policy \( \pi : S \rightarrow A \), which can maximize the cumulative reward for the recommender system

**Problem Statement**

- **Markov Decision Process (MDP)**
  - Browsing history
  - User’s feedback (skip, click, or purchase)
  - State transition from \( s \) to \( s' \)
  - Discount factor \( \gamma \in [0, 1] \)

- **State:** \( s = (s_1, ..., s_N) \) is the previous \( N \) clicked or purchased items,
- **Transition:** from \( s \) to \( s' \):
  - If the user skips the item, then \( s' = (s_1, ..., s_N) \)
  - If the user clicks/purchases the item, then \( s' = (i_1, ..., i_N) \)

- **Goal:** find a recommendation policy \( \pi : S \rightarrow A \), which can maximize the cumulative reward for the recommender system

**Basic DQN Model**

- **State:** \( s = (i_1, ..., i_N) \) is the previous \( N \) clicked or purchased items
- **Transition:** from \( s \) to \( s' \):
  - If the user skips the item, then \( s' = (s_1, ..., s_N) \)
  - If the user clicks/purchases the item, then \( s' = (i_1, ..., i_N) \)

- **Q-function:**
  \[
  Q(s, a) = E_{s'} \left[ r + \gamma \max_{a'} Q(s', a') | s, a \right]
  \]

- **Q-learning update:**
  \[
  Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') | s, a \right] - Q(s, a)
  \]

**The Proposed Framework**

- **The Architecture of DEERS**
  - State \( s = (s_1, s_2) \), where \( s_1 = (j_1, ..., j_N) \) is the previous \( N \) clicked or purchased items,
  - Transition from \( s \) to \( s' \):
    - If the user skips the item, then \( s' = (s_1, ..., s_N) \)
    - If the user clicks/purchases the item, then \( s' = (i_1, ..., i_N) \)

- **RNN with GRU to capture users’ sequential preference**
  - Recommend an item that is similar to the clicked/purchased items (left part), while dissimilar to the skipped items (right part)

- **The Pairwise Regularization Term**
  - RA often recommends items belong to the same category, while users click/purchase a part of them and skip others

**Experiment**

- **Dataset from JD.com**
- **Baselines**
  - Collaborative filtering
  - Factorization Machines
  - RNN with GRU
  - DEERS-p (only positive feedback)

- **It can be observed:**
  - CF and FM perform worse than GRU and DEERS, since CF and FM ignore the temporal sequence of the users’ browsing history
  - GRU performs worse than than DEERS-p, since GRU maximizes the immediate reward for recommendations
  - DEERS performs better than DEERS-p because DEERS integrates both positive and negative items (or feedback)

**Conclusion**

- **We design a novel architecture to capture both positive and negative feedback simultaneously**
- **We design a pairwise regularization term to maximize the difference of Q-values between competing items**
- **This work is supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940**