A. Lemma

A.1. Proof of lemma 1

Lemma 1. *Let the regularization term in Eq. (2) be $R(\phi) = \sum_{i=1}^{k} \phi_i \log \phi_i$. Then the optimal solution ϕ^* for the problem in Eq. (2) has a closed form*

$$\phi^*(\boldsymbol{s}^t, \mathcal{A}^t)_i = \exp(\eta r(\boldsymbol{s}^t, a_i)) / \textstyle\sum_{a_j \in \mathcal{A}^t} \exp(\eta r(\boldsymbol{s}^t, a_j)).$$

Furthermore, in each session t, the user's optimal policy ϕ^* is equivalent to the following discrete choice model where ε^t follows a Gumbel distribution.

$$a^t = \arg\max_{a \in \mathcal{A}^t} \eta\, r(\boldsymbol{s}^t, a) + \varepsilon^t. \tag{3}$$

Proof. First, recall the problem defined in Eq. (2):

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

Denote $\phi^t = \phi(\boldsymbol{s}^t, \mathcal{A}^t)$. Since ϕ can be an arbitrary mapping (i.e., ϕ is not limited in a specific parameter space), ϕ^t can be an arbitrary vector in Δ^{k-1}. Recall the notation $\mathcal{A}^t = \{a_1, \cdots, a_k\}$. Then the expectation taken over random variable $a^t \in \mathcal{A}^t$ can be written as

$$\mathbb{E}_\phi \left[r(\boldsymbol{s}^t, a^t) \right] - \frac{1}{\eta} R(\phi) = \sum_{i=1}^{k} \phi_i^t r(\boldsymbol{s}^t, a_i) - \frac{1}{\eta} \sum_{i=1}^{k} \phi_i^t \log \phi_i^t. \tag{12}$$

By simple computation, the optimal vector $\phi^{t*} \in \Delta^{k-1}$ which maximizes Eq. (12) is

$$\phi_i^{t*} = \frac{\exp(\eta r(\boldsymbol{s}^t, a_i))}{\sum_{j=1}^{k} \exp(\eta r(\boldsymbol{s}^t, a_j))}, \tag{13}$$

which is equivalent to Eq. (2). Next, we show the equivalence of Eq. (13) to the discrete choice model interpreted by Eq. (3).

The cumulative distribution function for the Gumbel distribution is $F(\varepsilon; \alpha) = \mathbb{P}[\varepsilon \leqslant \alpha] = e^{-e^{-\alpha}}$ and the probability density is $f(\varepsilon) = e^{-e^{-\varepsilon}} e^{-\varepsilon}$. Using the definition of the Gumbel distribution, the probability of the event $[a^t = a_i]$ where a^t is defined in Eq. (3) is

$$\begin{aligned} P_i := \mathbb{P}\Big[a^t = a_i\Big] &= \mathbb{P}\Big[\eta r(\boldsymbol{s}^t, a_i) + \varepsilon_i \geqslant \eta r(\boldsymbol{s}^t, a_j) + \varepsilon_j, \text{ for all } i \neq j\Big] \\ &= \mathbb{P}\Big[\varepsilon_j \leqslant \varepsilon_i + \eta r(\boldsymbol{s}^t, a_i) - \eta r(\boldsymbol{s}^t, a_j), \text{ for all } i \neq j\Big]. \end{aligned}$$

Suppose we know the random variable ε_i. Then we can compute the choice probability P_i conditioned on this information. Let $B_{ij} = \varepsilon_i + \eta r(\boldsymbol{s}^t, a_i) - \eta r(\boldsymbol{s}^t, a_j)$ and $P_{i|\mathcal{E}}$ be the conditional probability; then we have

$$P_{i|\varepsilon_i} = \prod_{i \neq j} \mathbb{P}[\varepsilon_j \leqslant B_{ij}] = \prod_{i \neq j} e^{-e^{-B_{ij}}}.$$

In fact, we only know the density of ε_i. Hence, using the Bayes theorem, we can express P_i as

$$\begin{aligned} P_i &= \int_{-\infty}^{\infty} P_{i|\varepsilon_i} f(\varepsilon_i) \mathrm{d}\varepsilon_i = \int_{-\infty}^{\infty} \prod_{i \neq j} e^{-e^{-B_{ij}}} f(\varepsilon_i) \mathrm{d}\varepsilon_i \\ &= \int_{-\infty}^{\infty} \prod_{j=1}^{k} e^{-e^{-B_{ij}}} e^{e^{-\varepsilon_i}} e^{-e^{-\varepsilon_i}} e^{-\varepsilon_i} \mathrm{d}\varepsilon_i = \int_{-\infty}^{\infty} \Big(\prod_{j=1}^{k} e^{-e^{-B_{ij}}} \Big) e^{-\varepsilon_i} \mathrm{d}\varepsilon_i \end{aligned}$$

Now, let us look at the product itself.

$$\begin{aligned} \prod_{j=1}^{k} e^{-e^{-B_{ij}}} &= \exp\Big(-\sum_{j=1}^{k} e^{-B_{ij}} \Big) \\ &= \exp\Big(-e^{-\varepsilon_i} \sum_{j=1}^{k} e^{-(\eta r(\boldsymbol{s}^t, a_i) - \eta r(\boldsymbol{s}^t, a_j))} \Big) \end{aligned}$$

Hence

$$P_i = \int_{-\infty}^{\infty} \exp(-e^{-\varepsilon_i} Q) e^{-\varepsilon_i} \mathrm{d}\varepsilon_i$$

where $Q = \sum_{j=1}^{k} e^{-(\eta r(\boldsymbol{s}^t, a_i) - \eta r(\boldsymbol{s}^t, a_j))} = Z / \exp(\eta r(\boldsymbol{s}^t, a_i))$.

Next, we make a change of variable $y = e^{-\varepsilon_i}$. The Jacobian of the inverse transform is $J = \frac{\mathrm{d}\varepsilon_i}{\mathrm{d}y} = -\frac{1}{y}$. Since $y > 0$, the absolute of Jacobian is $|J| = \frac{1}{y}$. Therefore,

$$\begin{aligned} P_i &= \int_0^{\infty} \exp(-Qy) y |J| \mathrm{d}y = \int_0^{\infty} \exp(-Qy) \mathrm{d}y \\ &= \frac{1}{Q} = \frac{1}{\exp(-\eta r(\boldsymbol{s}^t, a_i)) \sum_j \exp(\eta r(\boldsymbol{s}^t, a_j))} \\ &= \frac{\exp(\eta r(\boldsymbol{s}^t, a_i)}{\sum_{j=1}^{k} \exp(\eta r(\boldsymbol{s}^t, a_j))}. \end{aligned}$$

□

A.2. Proof of lemma 2

Lemma 2. *When $R(\phi) = \sum_{i=1}^{k} \phi_i \log \phi_i$, the optimization problem in Eq. (5) is equivalent to the following maximum likelihood estimation*

$$\max_{\theta \in \Theta} \prod_{t=1}^{T} \frac{\exp(\eta r_\theta(\boldsymbol{s}^t_{true}, a^t_{true}))}{\sum_{a^t \in \mathcal{A}^t} \exp(\eta r_\theta(\boldsymbol{s}^t_{true}, a^t))}.$$

Proof. This lemma is a straight forward result of lemma 1. First, recall the problem defined in Eq. (5):

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

We make a assumption that there is no repeated pair $(\boldsymbol{s}^t_{true}, a^t)$ in Eq. (5). This is a very soft assumption because $\boldsymbol{s}^t_{true}$ is updated overtime, and a^t is in fact representing its feature vector $\boldsymbol{f}^t_{a^t}$, which is in space $\mathbb{R}^d$. With this assumption, we can let ϕ map each pair $(\boldsymbol{s}^t_{true}, a^t)$ to the optimal vector ϕ^{t*} which maximize $r_\theta(\boldsymbol{s}^t_{true}, a^t) - \frac{1}{\eta} R(\phi^t)$ since there is no repeated pair. Using Eq. (13), we have

$$\begin{aligned} &\max_{\phi \in \Phi} \mathbb{E}_\phi \left[\sum_{t=1}^{T} r_\theta(\boldsymbol{s}^t_{true}, a^t) \right] - \frac{1}{\eta} R(\phi) = \max_{\phi \in \Phi} \sum_{t=1}^{T} \mathbb{E}_\phi \left[r_\theta(\boldsymbol{s}^t_{true}, a^t) \right] - \frac{1}{\eta} R(\phi) \\ =& \sum_{t=1}^{T} \left(\sum_{i=1}^{k} \phi_i^{t*} r(\boldsymbol{s}^t, a_i) - \frac{1}{\eta} \sum_{i=1}^{k} \phi_i^{t*} \log \phi_i^{t*} \right) = \sum_{t=1}^{T} \frac{1}{\eta} \log \Big(\sum_{i=1}^{k} \exp(\eta r_\theta(\boldsymbol{s}^t_{true}, a_i)) \Big). \end{aligned}$$

Eq. (5) can then be written as

$$\min_{\theta \in \Theta} \sum_{t=1}^{T} \frac{1}{\eta} \log \Big(\sum_{i=1}^{k} \exp(\eta r_\theta(\boldsymbol{s}^t_{true}, a_i)) \Big) - \sum_{t=1}^{T} r_\theta(\boldsymbol{s}^t_{true}, a^t_{true}),$$

which is the negative log-likelihood function and is equivalent to lemma 2. □

B. Alogrithm box

The following is the algorithm of learning the cascading deep Q-networks. We employ the cascading Q functions to search the optimal action efficiently (line 2). Besides, both the experience replay (Mnih et al., 2013) and ε-exploration techniques are applied. The system's experiences at each time-step are stored in a replay memory set $\mathcal{M}$ (line 2) and then a minibatch of data will be sampled from the replay memory to update $\widehat{Q^j}$ (line 2 and 2). An exploration to the action space is executed with probability ε (line 2).

Algorithm 2 cascading deep Q-learning (CDQN) with Experience Replay

Initialize replay memory $\mathcal{M}$ to capacity N

Initialize parameter Θ_j of $\widehat{Q^j}$ with random weights for each $1 \leq j \leq k$ **For** *iteration* $i = 1$ *to* L **do**

  Sample a batch of users $\mathcal{U}$ from training set Initialize the states $\boldsymbol{s}^0$ to a zero vector for each $u \in \mathcal{U}$ **For** $t = 1$ *to* T **do**

    For *each user* $u \in \mathcal{U}$ *simultaneously* **do**

      With probability ε select a random subset $\mathcal{A}^t$ of size k

      Otherwise, $\mathcal{A}^t = \text{ARGMAX\_Q}(\boldsymbol{s}_u^t, \mathcal{I}^t, \Theta_1, \cdots, \Theta_k)$

      Recommend $\mathcal{A}^t$ to user u, observe user action $a^t \sim \phi(\boldsymbol{s}^t, \mathcal{A}^t)$ and update user state $\boldsymbol{s}^{t+1}$

      Add tuple $\left(\boldsymbol{s}^t, \mathcal{A}^t, r(\boldsymbol{s}^t, a^t), \boldsymbol{s}^{t+1}\right)$ to $\mathcal{M}$

    Sample random minibatch $B \overset{\text{iid.}}{\sim} \mathcal{M}$

    For each j, update Θ_j by SGD over the loss $\left(y - \widehat{Q^j}(\boldsymbol{s}^t, A_{1:j}^t; \Theta_j)\right)^2$ for B

return $\Theta_1, \cdots, \Theta_k$

C. Dataset description

(1) MovieLens public dataset[1] contains large amounts of movie ratings collected from their website. We randomly sample 1,000 active users from this dataset. On average, each of these active users rated more than 500 movies (including short films), so we assume they rated almost every movie that they watched and thus equate their rating behavior with watching behavior. MovieLens dataset is the most suitable public dataset for our experiments, but it is still not perfect. In fact, none of the public datasets provides the context in which a user's choice is made. Thus, we simulate this missing information in a reasonable way. For each movie watched(rated) on the date d, we collect a list of movies released within a month before that day d. On average, movies run for about four weeks in theater. Even though we don't know the actual context of user's choice, at least the user decided to watch the rated movie instead of other movies in theater. Besides, we control the maximal size of each displayed set by 40. **Features:** In MovieLens dataset, only titles and IDs of the movies are given, so we collect detailed movie information from Internet Movie Database(IMDB). Categorical features as encoded as sparse vectors and descriptive features are encoded as dense vectors. The combination of such two types of vectors produces 722 dimensional raw feature vectors. To further reduce dimensionality, we use logistic regression to fit a wide&deep networks (Cheng et al., 2016) and use the learned input and hidden layers to reduce the feature to 10 dimension.

(2) An online news article recommendation dataset from Ant Financial is anonymously collected from Ant Financial news article online platform. It consists of 50,000 users' clicks and impression logs for one month, involving dozens of thousands of news. It is a time-stamped dataset which contains user features, news article features and the context where the user clicks the articles. The size of the display set is not fixed, since a user can browse the news article platform as she likes. On average a display set contains 5 new articles, but it actually various from 2 to 10. **Features:** The news article raw features are approximately of dimension 100 million because it summarizes the key words in the article. Apparently it is too expensive to use these raw features in practice. The features we use in the experiments are 20 dimensional dense vector embedding produced from the raw feature by wide&deep networks. The reduced 20 dimensional features are widely used in this online platform and revealed to be effective in practice.

(3) Last.fm[2] contains listening records from 359,347 users. Each display set is simulated by collecting 9 songs with nearest time-stamp.

(4) Yelp[3] contains users' reviews to various businesses. Each display set is simulated by collecting 9 businesses with nearest location.

(5) RecSys15[4] contains click-streams that sometimes end with purchase events.

(6) Taobao[5] contains the clicking behavior and buying behavior of users in 22 days. We consider the buying behaviors as positive events.

[1] https://grouplens.org/datasets/movielens/

[2] https://www.last.fm/api

[3] https://www.yelp.com/dataset/

[4] https://2015.recsyschallenge.com/

[5] https://tianchi.aliyun.com/datalab

D. More figures for experimental results

D.1. Figures for section 6.1

An interesting comparison is shown in Figure 3 and more similar figures are provided here. The blue curve is the trajectory of a user's actual choices of movies over time. The orange curves are simulated trajectories predicted by GAN and CCF, respectively. Similar to what we conclude in section 6.1, these figures reveal the good performances of GAN user model in terms of capturing the evolution of users' interest.

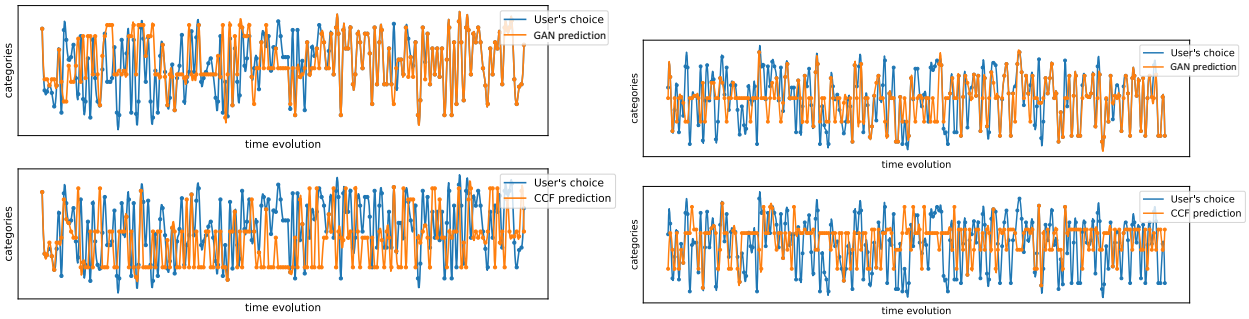

Figure 7. Two more examples: comparison of the true trajectory(blue) of user's choices, the simulated trajectory predicted by GAN model (orange curve in upper sub-figure) and the simulated trajectory predicted by CCF (orange curve in the lower sub-figure) for the same user. Y-axis represents 80 categories of movies.

D.2. Figures for section 6.2

We demonstrate the policy performance in user level in figure 4 by comparing the cumulative reward. Here we attach the figure which compares the click rate. In each sub-figure, red curve represents GAN-DQN policy and blue curve represents the other. GAN-DQN policy contributes higher averaged click rate for most users.

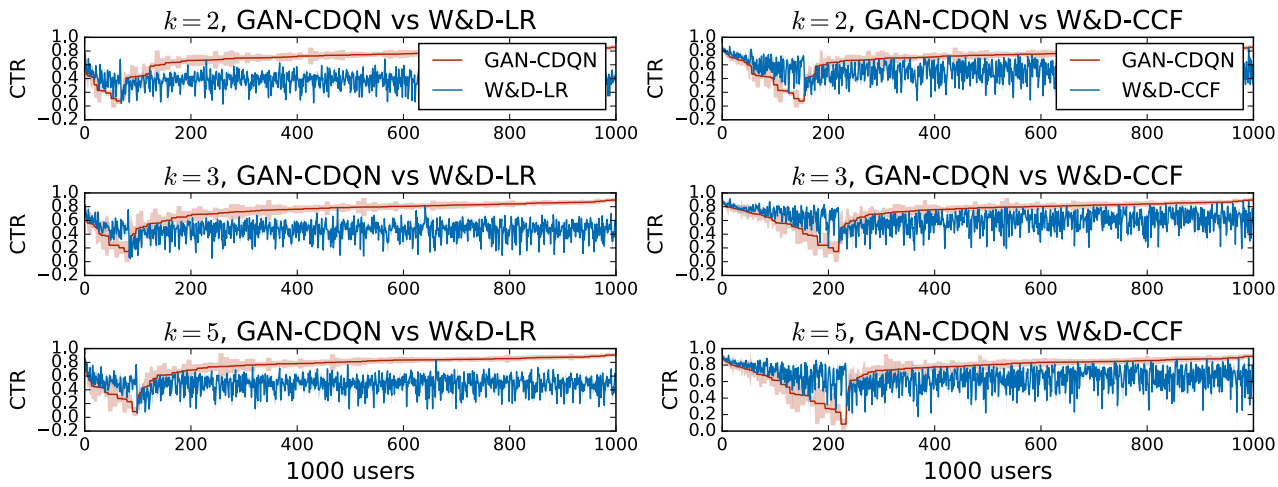

Figure 8. Comparison of click rates among 1,000 users under the recommendation policies based on different user models. In each figure, red curve represents GAN-DQN policy and blue curve represents the other. The experiments are repeated for 50 times and standard deviation is plotted as the shaded area. This figure is similar to figure 4, except that it plots the value of click rates instead of user's cumulative rewards.

D.3. Figures for section 6.3

This figure shows three sets of results corresponding to different sizes of display set. It reveals how users' cumulative reward(averaged over 1,000 users) increases as each policy interacts with and adapts to 1,000 users over time. It can be

easily that the CDQN policy pre-trained over a GAN user model can adapt to online users much faster then other model-free policies and can reduce the risk of losing the user at the beginning. The experiment setting is similar to section 6.2. All policies are evaluated on a separated set of 1,000 users associated with a test model. We need to emphasize that the GAN model which assists the CDQN policy is learned from a training set of users without overlapping test users. It is different from the test model which fits the 1,000 test users.

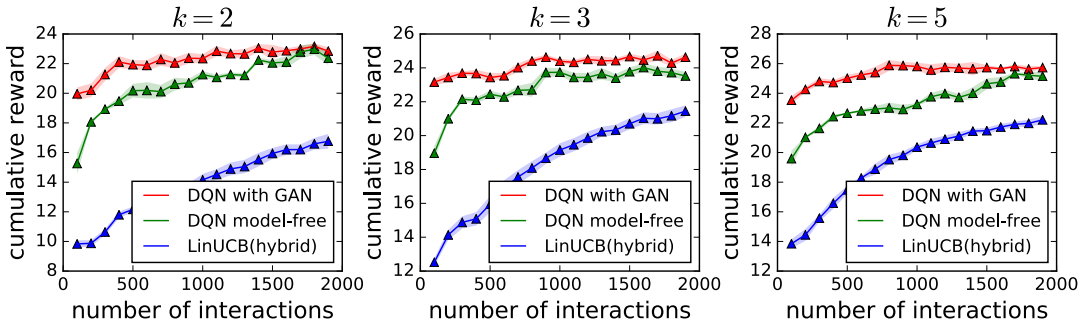

Figure 9. Comparison of the averaged cumulative reward among 1,000 users under different adaptive recommendation policies. X-axis represents how many times the recommender interacts with online users. Here the recommender interact with 1,000 users each time, so in fact each interaction represents 100 online data points. Y-axis is the click rate. Each point (x, y) in this figure means a click rate y is achieved after x many times of interactions with the users.