Online Appendix

A. DRL Algorithms Details

A.1. Q-learning vs. DQN

In Q-learning, we defined the optimal action-value function as $Q^*(s,a) = max_\pi Q^\pi(s,a)$, which has a maximum expected return when following the optimal policy. $Q^*(s,a)$ should follow the *Bellman equation* (Bellman 2013):

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a')|s,a\right] \tag{1}$$

where s', a' are the next state and action, respectively.

This solution logic works well with a countable state and action space. However, in reality, the state and marketing action space is often enormous and continuous, and as such, estimating the $Q^*(s,a)$ using the Bellman equation is infeasible as its problem complexity suffers from the curse of dimensionality. In other words, if the combinations of states and actions are too large, the memory and the computation requirement for Q will be too high. In addition, many state-action pairs may not appear in the real trace such that it is hard to update their values. Therefore Mnih et al. (2015) first proposed the use of a deep neural network as an approximator function to estimate the action-value function, i.e., $Q^*(s,a) \approx Q(s,a;\theta)$, where θ is the parameters of the deep neural network.

A.2. Experience Reply

A key ingredient behind the success of DQN is experience replay[1]. During the interacting with the consumer simulator, the AI agent retains a experience buffer[2] $\mathcal{D} = \{e_1, e_2, ..., e_t\}$ in which to store previous experiences, where e_t is the (s,a,r,s') at time step t. Instead of training Q-net using the current experience, the network was trained by sampling mini-batches of experience from $\mathcal{D}$ uniformly at random. The sequence of losses thus takes the form at iteration i:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{U}(\mathcal{D})}\left[(Q_i^{target} - Q_{\theta_i}(s,a))^2\right] \tag{2}$$

This experience reply strategy can reduce variances as uniform sampling from the reply buffer reduces the correlations among samples used in the update, and thus leads to stable Q-net training.

B. DRL implementation algorithms

The overall algorithm is summarized in Algorithm 1. We first initialized the three important hyper-parameters k, β and γ, and set the hash table of counting the state-action pairs with 0. Then, we loaded the trained consumer behavior model with LSTM as an environment to interact with AI agents. For each learning iteration, we randomly sampled the 2000 initial states from our data at the same time to represent 2000

[1] Instead of training Q-net using the current and single experience, the network was trained by sampling mini-batches of experience from a container uniformly at random

[2] A container to hold the previous experience (e.g., the previous transitions (s,a,r,s')) during the interaction with consumer model

consumers, DRL agents obtained the initial state and converted the states to the bit-wise bucket and checked the previous action counts at this bucket. Then these action counts were converted to bonus–$B(s,a)$, the agent chose an action maximizing $[Q_\theta(s,a)+B(s,a)]$. When the new action for this state was chosen, we set this new state bucket-action pair to the hash table. Further, this action would be passed to the consumer model to get the feedback from the consumer side including reward r and next states s'. Now one round of interaction was completed and we got one transition experience (s,a,r,s'), we then stored this experience to experience buffer. s' would be the new current state to continue for the next round of interaction. The interaction for each consumer would last 30 days. When 30 rounds of interactions were completed, one learning iteration was completed. The parameters of online Double Dueling DQN agents were updated each consumer interaction, the parameters of the target net were replaced with an online net every 100 learning iterations. We trained 30,000 epochs for each agent with predefined k, β and γ combination.

Algorithm 1 Dueling Double DQN with Proposed Risk-seeking Exploration Strategy

Initialize k, β and γ
Initialize the hash table to count the state bucket-action pair $N(\psi(s),a)$ with 0
Load consumer behavior model
for $i=1$ to *iterations* **do**
  Random sample consumer initial states s_1 from our data
  for $t=1$ to *Episodes* **do**
    Calculate the state bucket and check previous action counts $N(\psi(s),a)$ from hash table
    Choose action a_t maximizing $[Q_\theta(s_t,a_t)+B(s_t,a_t)]$
    Set the new state bucket-action pair to the hash table e.g. $N(\psi(s_t),a_t)=N(\psi(s_t),a_t)+1$
    Interact with consumer model to get reward r_t and next states s_{t+1}
    Store transtions(s_t,a_t,r_t,s_{t+1}) to the experience buffer
    Update Dueling Double DQN's parameters
    $s_t = s_{t+1}$
  end for
end for

C. Simulator training details

To train these consumer dynamic behavior models from our data, we took one consumer sequential states and marketing actions from our randomized experiment data as example. We denoted the input time series as $X=\{x_1,x_2,...,x_t\}$. Doing a forward pass with any model (e.g., Multi-task LSTM) using this consumer's sequential information, we had a predicted state and reward sequence $\widehat{S}=\{\widehat{s}_1,\widehat{s}_2,...,\widehat{s}_t\}$ and $\widehat{R}=\{\widehat{r}_1,\widehat{r}_2,...,\widehat{r}_t\}$, where we had predicted value at each time step. The ground truth $S=\{s_1,s_2,...,s_t\}$ and $R=\{r_1,r_2,...,r_t\}$ gathered using the actual next state and actual reward from our data. The $\text{MSE}_{\text{Next state prediction}}$ was calculated

based on predicted $\widehat{S}$ and ground truth S for each consumer sequence in our training data. Same process for $\text{MSE}_{\text{Reward prediction}}$.

$$\text{MSE}_{\text{Next state prediction}} = \sum_{i=1}^{N} l\left(S_i, \hat{S}_i^{(p)}\right) \tag{3}$$

$$\text{MSE}_{\text{Reward prediction}} = \sum_{i=1}^{N} l\left(R_i, \hat{R}_i^{(p)}\right) \tag{4}$$

where N is number of consumers.

D. Baseline models hyper-parameter tuning process

(1) Non-personalized mass promotions. No personal information was available for the agent to use. Therefore, all consumers received the same targeting policy. For this category, we implemented three policies:

- *Mass policy 1*: the manager applied promotion action every few days ψ for every consumer; the promotion action was randomly selected from price and free-content promotions with equal probability. The experiment was performed with $\psi \in \{1, 2, 3, 4, 5\}$. Then, the best performance was selected as the basis of the comparison.
- *Mass policy 2*: the manager applied promotion action every few days τ for every consumer while the price and content promotion took turns to be used. The experiment was performed with $\tau \in \{1, 2, 3, 4, 5\}$. Then, the best performance was selected as the basis of comparison.
- *Random policy*: the manager randomly picked up a certain percentage (ζ) of consumers to use price promotion and a certain percentage (η) of consumers to use content promotion for every day. The experiment was performed with $\zeta, \eta \in \{10\%, 20\%, 30\%, 40\%, 50\%\}$. Then, the best performance was selected as the basis of comparison.

(2) Myopic personalized targeting. These myopic methods did not consider expected future rewards when deciding at any point. The objective function for myopic policies is:

$$a = \arg\max_{a \in \mathcal{A}} R_t(s_{i,t}, a) \tag{5}$$

In these myopic methods, the promotion action maximizing the immediate revenue in the current period was selected by the platform. At each time step, the time-varying promotion action maximizing the immediate reward was selected based on the time-varying state variable. However, these methods were myopic considering that it ignored the future revenue when making decisions. Our DRL selected an action maximizing the sum of immediate and expected future revenue at each time step, that is, $a = \arg\max_{a \in \mathcal{A}} Q(s, a)$, where $Q(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q(s', a') | s, a\right]$. Here, five different solutions documented in the literature were provided to achieve this myopic objective:

- *XGBoost*: the state-action interaction (s, a) with immediate reward (r) from data was empirically estimated using extreme gradient boosting (Chen and Guestrin 2016), which was a fast and scalable version of Boosted Regression Tree (Friedman 2001). XGBoost has been verified to outperform most existing machine learning methods in most prediction contests, especially those related to consumer decision-making such as that proposed in this paper (Chen and Guestrin 2016). For a given transition $(s_{i,t}, a_{i,t}, R_{i,t})$ in our data for consumer i at time step t, XGBoost applied K additive functions to predict the output.

$$\hat{R}_{i,t}(s_{i,t}, a_{i,t}) = \phi(s_{i,t}, a_{i,t}) = \sum_{k=1}^{K} f_k(s_{i,t}, a_{i,t}), \quad f_k \in \mathcal{F} \tag{6}$$

where $\mathcal{F} = \{f(s_{i,t}, a_{i,t}) = w_{q(s_{i,t}, a_{i,t})}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ denotes the space of regression trees (also known as CART); q represents the structure of each tree that mapped an example to the corresponding leaf index; T refers to the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights w. Then, the following regularized objective in iteration p was minimized to learn the set of parameters in the model.

$$\mathcal{L}^{(p)} = \sum_{i=1}^{n} \sum_{t=1}^{T_i} l\left(R_{i,t}, \hat{R}_{i,t}^{(p+1)} + f_h(s_{i,t}, a_{i,t})\right) + \Omega(f_p) \tag{7}$$

After the trained model $\hat{R}_{i,t}(s_{i,t}, a_{i,t})$ was obtained, the test on our simulator was conducted. Our dataset had 12,160 consumer initial states; for each state, the agent selected the action based on the policy:

$$\pi(a|s) = \begin{cases} 1 & a = \arg\max_{a \in \mathcal{A}} \hat{R}_t(s_{i,t}, a) \\ 0 & otherwise \end{cases} \tag{8}$$

Then, the consumer model provided the generated reward and the next state for the next round of action was selected by the agent.

- *Engagement-stage-based targeting*: engagement-stage-based method (Zhang et al. 2019) divided the consumers' reading behaviors into four groups "aware, exploring, active, addicted" states. It was discovered that the aware consumers who were the least familiar with the app preferred price promotion while the addicted consumers had more interest in the free-content promotion. We applied price promotion to addicted states according to their strategy while applying the content promotion to the aware and exploring states, and no promotion was applied to active states.

We used k-means to partition all the states ($s_{i,t}$) in historical data into four clusters $C = \{C_1, C_2, C_3, C_4\}$;the four clusters were named as aware, exploring, active, and addicted according to the centroid's values. A consumer started with the initial state; then, the distance between the state and four groups' centroids was calculated; next, the state was labeled as the nearest cluster. Additionally, a promotion action was selected depending on the following policy:

$$a_{i,t} = \begin{cases} price\ \ promotion, & if\ \ s_{i,t} \in C_1 \\ free-content\ \ promotion, & if\ \ s_{i,t} \in C_4 \\ No\ \ promotion, & otherwise \end{cases} \tag{9}$$

- *Past-experience-based targeting*: past-experience-based targeting strategy (Mooney and Roy 2000, Fudenberg and Villas-Boas 2006) is similar to the industry's current effort wherein an app company monitors consumers' past-purchase records. We used consumers' daily historical purchase amount ($R_{i,t}$) to divide them into four quantiles $q = \{q_1, q_2, q_3, q_4\}$. The price and content promotion were only applied in certain quantiles. For each day targeting, an agent looked at the previous day's purchase amount and applied the corresponding targeting. Consumers' initial state and initial reward are sampled from our dataset. Afterward, the previous reward $R_{i,t-1}$ was tracked to select a promotion action $a_{i,t}$ for each consumer i at time step t,

observing which quantiles the historical purchase behavior fall in $q_j, j \in \{1,2,3,4\}$; besides, the corresponding promotion action was applied. The experiment was conducted with every possible combination. Then, the best performance was selected as the basis of comparison.

- *Similar-taste-based*: this approach is similar to collaborative filtering(Linden et al. 2003, Breese et al. 1998), the most popular and widely used personalized recommendation technique. It assumed that people typically obtained the best recommendations from people with similar tastes to themselves. We followed this approach and divided the consumers ($s_{i,t}$) into different groups using historical data $g = \{g_1, g_2, ..., g_K\}$, where k was the group numbers to be tuned by experiments; meanwhile, consumers with similar reading behaviors were in the same group. The preferred actions with the highest average historical reward for each group were calculated to observe the taste of each group. Then, which group preferred which action was revealed. A consumer randomly started with the initial state ($s_{i,t}$); then, the distance between the state and all groups' centroids were calculated; the state was labeled as the nearest groups. Next, the preferred action of its group was applied to this consumer at the current time step. A promotion action $a_{i,t}$ based on state $s_{i,t}$ was selected according to the following policy:

$$a_{i,t} = \arg\max_{a \in \mathcal{A}} \frac{1}{\mid g_k \mid} \sum_{\tilde{s} \in g_k} R(\tilde{s}, a) \qquad if \quad s_{i,t} \in g_k \tag{10}$$

where $k \in \{1, 2, ..., K\}$. The experiment was conducted with $K \in \{3, 4, 5, 6\}$. Then, the best performance was select as the basis of comparison.

- *Heterogeneous treatment effect by orthogonal Random Forest (Oprescu et al. 2018)*: performing a regression of the outcome variable (R) on the treatment (a) and state variable (s) with XGBoost can be an indirect and empirical solution to myopic targeting, because it mixed up the effect of treatment (i.e., promotion action) and the direct effect of the state on outcomes. Instead, there was a direct solution: *heterogeneous treatment effect* (HTE) estimation. HTE estimation was a fundamental problem in causal inference from observational data and at the core of many decision-making processes, including clinical trial assignments to patients, price adjustments of productions, and ad recommendation. Specifically, the heterogeneity of the population was considered to estimate HTE, namely, the effect of a price promotion/free-content promotion treatment T on the outcome Y of interest (such as revenue); it was regarded as a function of observable characteristics x (consumer behavior features) of the treated subject (such as consumer). Orthogonal Random Forest (Oprescu et al. 2018) was an orthogonalized version of causal forest (i.e, Generalized Random Forest) which was more robust to the nuisance estimation error. Considering the observations $D = \{Z_i = (a_i, s_i, R_i)\}$, where the state was a feature vector capturing the heterogeneity (s), and action represents the discrete treatments (no promotion, price promotion and free-content promotion) ($a \in \{0, 1, 2\}$), and the immediate reward (R), the set of variables were related by the following equations:

$$R = \theta_j(s) * a + f_0(s) + \varepsilon, \quad j \in 1, 2 \tag{11}$$

where $\theta_j(s)$ denotes the heterogeneous effect of the treatment j on the outcome (R) as a function of the state (s): $j = 1$ stands for the HTE of price promotion over no promotion; $j = 2$ stands for the HTE of free-content

promotion over no promotion; $f_0(s)$ refers to an unknown nuisance function capturing the direct effect of the s on the outcome. ε represents the unobserved noises such that $E[\varepsilon|s,a]=0$. Our goal was to estimate the heterogeneous effect of the treatment (i.e., promotions action) on the outcome, $\theta(s)$.

The classical residualization approach was followed to create the orthogonal moment for identifying $\theta_j(s)$. By defining the function $f_0(s)=E[R|s]$ and considering the residuals – $\tilde{R}=R-f_0(s)$, the equation was simplified as

$$\widetilde{R}=\theta_j(s)*a+\varepsilon \tag{12}$$

We estimated $f_0(s)$ by using a forest lasso between R and s only among the subset of samples that received treatment $a=0$, which was no promotion action. For each other treatment price promotion or free-content promotion, a forest lasso between $\tilde{R}$ and s among the subset that received treatment $a=1$ or $a=2$ was performed to estimate the HTE – $\theta_j(s)$.

After the estimated HTE was obtained, a promotion action was selected according to the following policy to implement the personalized promotion action recommendation based on the HTE:

$$a_{i,t}=\begin{cases} price\ \ promotion, & if\ \ \theta_1(s_{i,t})>0\ \&\ \theta_1(s_{i,t})>\theta_2(s_{i,t}) \\ free-content\ \ promotion, & if\ \ \theta_2(s_{i,t})>0\ \&\ \theta_2(s_{i,t})>\theta_1(s_{i,t}) \\ No\ \ promotion, & otherwise \end{cases} \tag{13}$$

- *Multi-Armed Bandits* (MAB): this approach (Hauser et al. 2009, Schwartz et al. 2017, Urban et al. 2014) did not model the future revenue, as it assumed state did not change, and there was only action and reward interaction. In our approach, the future revenue can be modeled by estimating state transition, and the agent can observe the next state under a rough estimation of the future condition. In contrast, the MAB agent can only observe the reward. The MAB agent's objective was to learn the arm reward distribution, leading to finding the arm with the highest mean reward. MAB can be regarded as stateless or one-step of our approach. MAB assumed that the reward for each action was IID. For each action a, a distribution $\mathcal{D}_a$ over reals called the reward distribution exists. The reward was sampled independently from this distribution every time this action was selected. The reward distributions were initially unknown to the algorithm. The agents was mainly interested in the mean reward vector μ, where $\mu(a)=\mathbb{E}[\mathcal{D}_a]$ was the mean reward of the arm a. In this study, it was assumed that rewards followed the Gaussian distribution. The mean rewards were estimated according to the past experience by averaging the rewards associated with the target action a that we have observed so far (up to the current time step t):

$$\mu_t(a)=\frac{1}{K_t(a)}\sum_{i=1}^{t-1} r_i\,\mathbb{1}(a_i=a) \tag{14}$$

where $K_t(a)=\sum_{i=1}^{t-1}\mathbb{1}(a_i=a)$ was the number of times a taken prior to t; the right-hand side part was the sum of rewards when a is taken prior to t. The sample-average estimated converge to the true values if the action is taken an infinite number of times. A ϵ-greedy exploration-exploitation heuristic was adopted in this experiment. For any $\epsilon\in[0,1]$, the policy randomly allocated ϵ of the consumers allocated uniformly across the N promotions and allocates $1-\epsilon$ of consumers to the promotions with the highest empirical mean, namely,

the best action based on the agent's current knowledge. The allocation cross N promotion action was ϵ/N for all actions except for a^*, which had $\epsilon/N + (1-\epsilon)$. The experiment was conducted with $\epsilon \in \{10\%, 15\%, 20\%\}$. Then, the best performance was selected as the basis of comparison. This approach followed the policy:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{N} + (1-\epsilon) & for \;\; a^* = \arg\max_{a \in \mathcal{A}} \hat{\mu}(a) \\ \frac{\epsilon}{N} & otherwise \end{cases} \tag{15}$$

(3) Various exploration-exploitation heuristics. We also compared with various heuristics in other DRL settings.

- Greedy: this was a policy allocating all consumers to the promotion action with the largest Q at every decision period. It reflected pure exploitation without exploration:

$$\pi(a|s) = \begin{cases} 1 & for \;\; a^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_\theta(s,a) \\ 0 & otherwise \end{cases} \tag{16}$$

- ϵ-greedy: this was a randomized policy combining exploitation with a predetermined amount of exploration. For any $\epsilon \in [0,1]$, the policy randomly allocated ϵ of the observations allocated uniformly across the N promotions, and allocated $1-\epsilon$ of observations to the promotions with the highest Q value, that is, the best action based on the agent's current knowledge. The allocation cross N promotion action was ϵ/N for all actions except for a^*, namely, the best action that maximize the Q-net with $\epsilon/N + (1-\epsilon)$. The experiment was conducted with $\epsilon \in \{10\%, 15\%, 20\%\}$. This approach followed the policy:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{N} + (1-\epsilon) & for \;\; a^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_\theta(s,a) \\ \frac{\epsilon}{N} & otherwise \end{cases} \tag{17}$$

- Decaying ϵ-greedy: this strategy was based on the ϵ-greedy strategy, however the ϵ value decayed over time. In practice this meant that the strategy started out with a high ϵ, and thus a high exploration rate. Over time this ϵ grown ever smaller until it faded, optimally as the policy has converged, so that an optimal policy can be executed without having to take further (possibly sub optimal) exploratory actions.

$$\epsilon_t = \epsilon_{min} + (\epsilon_{initial} - \epsilon_{min}) * e^{-Decay\ rates * \#Steps} \tag{18}$$

where $\epsilon_{initial} \in \{1, 0.99, 0.95, 0.9\}$, $\epsilon_{min} \in \{0.15, 0.1, 0.01\}$, $Decay\ rates \in \{0.0005, 0.005, 0.05\}$

- Adaptive ϵ-greedy exploration based on value difference (VDBE) (Tokic 2010): decreasing-ϵ greedy considered "time" to reduce the exploration probability, VDBE was the advanced version which extends the decreasing-ϵ based on the true learning progress. Intuitively, on the one hand, the agent should explore more at the beginning of the learning process, which was recognized as large changes in value functions. On the other hand, the exploration rate should be reduced as agents gain more knowledge about the environment, which can be recognized as very small or no changes in the value function. This algorithm adjusted the exploration rate $\epsilon(s)$ based on the following rules:

$$\epsilon_{t+1}(s) = \delta f(s_t, a_t, \sigma) + (1-\delta)\epsilon_t(s) \tag{19}$$

$$f(s,a,\sigma) = \frac{1-e^{\frac{-|Q_{t+1}(s,a)-Q_t(s,a)|}{\sigma}}}{1+e^{\frac{-|Q_{t+1}(s,a)-Q_t(s,a)|}{\sigma}}} = \frac{1-e^{\frac{-|\alpha TD-Error|}{\sigma}}}{1+e^{\frac{-|\alpha TD-Error|}{\sigma}}} \tag{20}$$

where $\delta \in [0,1)$ is a hyper-parameter determining the influence of the selected action on the exploration rate. As suggested in (Wu et al. 2018), we set $\delta = \frac{1}{|\mathcal{A}|}$. We experimented with $\sigma \in \{0.1, 1, 5, 10, 20\}$

• Boltzmann exploration: this was an approach, in which the probability of selecting any action in a given state was determined by a softmax function applied to the predicted Q-values. Affinity for exploration is parameterized by the Boltzmann temperature ($T \in [0, +\infty]$). The agent tended to pure exploitation as $T \to 0$. Besides, the resulting distribution approached the uniform distribution (random exploration) as $T \to +\infty$. T can be gradually decreased over time to decrease exploration. This method performed well if the best action was well-separated from the others. However, it suffered somewhat when the values of the actions were close. The experiment was conducted with $T \in \{1, 10, 100, 1000\}$.

$$V_T = exp(\frac{x}{T}) \tag{21}$$

$$\pi(a|s) = \begin{cases} \frac{V_T(Q(s,a))}{\sum_{a \in A} V_T(Q(s,a))} & for \;\; a \in A \\ 0 & otherwise \end{cases} \tag{22}$$

• Decaying Boltzmann exploration: we gradually decreased the temperature T over time in order to decrease exploration.

$$T_t = T_{min} + (T_{initial} - T_{min}) * e^{-Decay\ rates * \#Steps} \tag{23}$$

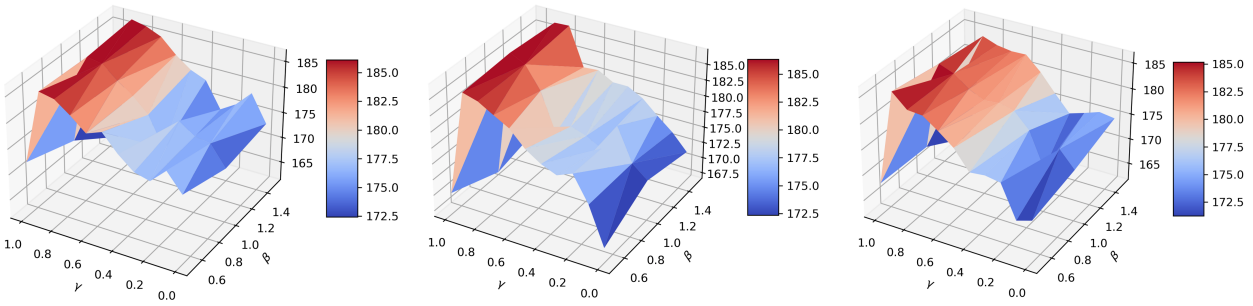
where $T_{initial} \in \{1000, 500, 100\}$, $T_{min} \in \{1, 0.1, 0.01\}$, $Decay\ rates \in \{0.0005, 0.005, 0.05\}$

• Boltzmann-Gumbel exploration (Cesa-Bianchi et al. 2017): this approach was an advanced version of Boltzmann exploration. This approach used different learning rates for different arms. Specifically, their approach was based on the observations that the distribution of $p_{t,i} \propto exp(\frac{1}{T}\hat{Q}_{t,i})$ can be equivalently specified by the rule $I_t = \arg\max_j \{\frac{1}{T}\hat{Q} + Z_{t,j}\}$, where $Z_{t,j}$ is a standard Gumbel random variable drawn independently for each arm j. This algorithm operated by independently drawing perturbations $Z_{t,i}$ from a standard Gumbel distribution for each arm i, then choosing action

$$I_{t+1} = \arg\max_i \{\hat{Q}_{t,i} + \beta_{t,i} Z_{t,i}\} \tag{24}$$

This approach used the simple choice $\beta_{t,i} = \sqrt{C^2/N_{t,i}}$ with the constant $C = 1/2$ as suggested by (Cesa-Bianchi et al. 2017).

• Count-based exploration (Tang et al. 2017): our heuristic is an extension of this approach. We shared a similar intuition that agent will get additional bonus when encountering less-visited situations. It differed from our approach in how the bonus is constructed and the incentive rule. In terms of incentive rule (i.e., how the bonus is used), Tang et al. (2017) adopted an indirect incentive rule. The bonus was used as additional reward beyond the original immediate reward. The bonus took effects through model parameter update with



(a) $k = 3$ (b) $k = 4$ (c) $k = 5$

Figure 1 What is the best extent of forward-looking? Cumulative rewards under different model specifications – the extent of forward-looking (γ), quantization granularity (k), and bonus coefficient (β)

new constructed total rewards $(r + r^+)$. However, such an indirect rule might have delayed effect, since the DRL algorithm usually adopted the experience replay strategy and it might take many rounds to make the bonus drive the agent's action. Different from it, we adapted to a direct incentive rule to mitigate such a delay effect. We use bonus in objective function $a = \arg\max_{a \in \mathcal{A}}[Q_\theta(s,a) + B(s,a)]$. In this way the bonus can directly affect the agent's action selection at each time step. Intuitively our proposed extension can explore faster than Tang et al. (2017).

This approach constructed bonus as inverse weight of state counts :

$$r^+(s) = \frac{\beta}{n(\psi(s))} \tag{25}$$

where ψ is the SimHash to quantize the continuous states to discrete buckets. where β is the bonus coefficient. This bonus cannot be used in our proposed direct incentive rule, since the bonus is only a function of state. If we added such a bonus to $Q(s,a)$, the bonus cannot affect the action recommendation since the bonus was not a function of action. Thus, we adapted the bonus as inverse weight of state-action pairs counts $B(s,a) = \frac{\beta}{(n(\psi(s),a))}$. The bonus can be added to the $Q(s,a)$ to directly affect agent's decision at each time step. We added this approach as baseline to demonstrate the benefit of adaptations through policy outcome and learning efficiency. We experimented with $k = \{3,4,5\}$ where k is the SimHash granularity, $\beta = \{0.5, 1, 1.5\}$.

E. Analysis about robustness of quantization granularity and bonus coefficient

Besides, we can observe large variations to the choices of quantization granularity (k) and bonus coefficient (β). For example, compared to k =3, the long-term reward is more curved for $k = 4$ and $k = 5$. This indicated with smaller quantization granularity; the model was less sensitive to γ. For the bonus coefficient, if we took the Figure 1c with fixed k, we found for every fixed γ, there was also a concave curve for (β). The performance increased from $\beta = 0.6$ to 1 and then decreased from β= 1 to 1.4. This indicated too little or too much bonus was worse than the right bonus coefficient. β and k were hyperparameters that we needed to tune to have the best performance.

F. Online testing baseline training and testing process

Specifically, for three categories baseline approaches:

- Non-personalized mass promotions: we found the optimal percentage on the simulated training environment and deployed the optimal policy on simulated testing environment.

• Myopic non-exploration/exploitation approaches: we used training samples of simulated training environment to learn the fixed optimal policy and tested the learned optimal policy on the simulated testing environment.

• DRL approaches and myopic exploration/exploitation approaches (i.e., MAB): We trained on the simulated training environment, tested on simulated testing environment.

G. Policy evaluation on individual simulators for main results and simulated online testing

For main results, five individual simulator results were shown in Table 1. For simulated online testing, five individual simulator results were shown in Table 2.

Category	Model	Revenue on various simulators					Average	Std	Improvement
		Multi-task LSTM	Single-task LSTM	Multi-task GRU	Multi-task RNN	Multi-task MLP			
Non-personalized mass promotions	Mass policy 1	142.21	139.84	143.51	144.78	145.42	143.15	2.22	29.97%
	Mass policy 2	140.14	137.45	136.85	140.41	141.48	139.26	2.00	33.60%
	Random policy	141.89	137.87	141.45	143.74	143.47	141.68	2.34	31.32%
Myopic personalized targeting	XGBoost	150.48	154.41	152.32	151.41	153.74	152.47	1.61	22.02%
	Engagement-stage-based	147.74	145.57	146.84	148.48	148.84	147.49	1.32	26.14%
	Past-experience-based	145.57	146.74	144.45	146.68	150.24	146.73	2.17	26.79%
	Similar-taste-based	144.54	145.78	146.87	147.41	145.46	146.01	1.14	27.42%
	HTE by ORF	151.75	152.48	154.84	151.75	152.48	152.66	1.27	21.87%
	Multi-armed bandit	152.75	150.73	155.57	151.54	154.87	153.09	2.08	21.53%
Various exploration-exploitation heuristic	Greedy	165.42	165.86	167.49	164.42	168.84	166.40	1.75	11.81%
	$\epsilon - greedy$	174.12	172.75	171.47	173.48	175.57	173.47	1.52	7.25%
	Decaying $\epsilon - greedy$	178.47	176.48	177.85	176.42	175.98	177.04	1.06	5.09%
	VDBE	177.85	174.85	176.75	177.45	177.85	176.95	1.25	5.14%
	Boltzmann	172.42	169.54	170.5	170.48	171.98	170.98	1.18	8.81%
	Decaying Boltzmann	174.75	175.85	174.42	172.84	172.42	174.05	1.41	6.89%
	Boltzmann-Gumbel	179.42	179.59	177.25	175.82	178.42	178.1	1.58	4.46%
	Count-based exploration	180.41	182.54	184.36	179.51	180.92	181.54	1.91	2.5%
	Quantization-based uncertain learning	186.74	187.48	185.34	184.48	186.26	186.06	1.17	-
DQN variants	Noisy net	170.48	173.85	174.96	171.75	174.45	173.09	1.90	7.48%
	Prioritized replay	175.57	174.85	177.12	173.54	176.65	175.54	1.43	5.98%

Table 1 Main Results – DRL and Baselines Model Comparison with Top-5 Simulators (Individual Simulator Results)

Category	Model	Revenue on various simulators					Average	Std	Improvement
		Multi-task LSTM	Single-task LSTM	Multi-task GRU	Multi-task RNN	Multi-task MLP			
Non-personalized mass promotions	Mass policy 1	134.17	135.65	131.03	139.06	132.89	134.56	3.03	34.18%
	Mass policy 2	136.42	132.48	137.18	134.25	137.48	135.56	2.13	33.18%
	Random policy	137.18	130.72	135.98	135.48	132.09	134.29	2.74	34.45%
Myopic personalized targeting	XGBoost	145.42	147.48	143.08	150.41	148.47	146.97	2.82	22.84%
	Engagement-stage-based	141.85	143.7	147.43	145.82	140.41	143.84	2.85	25.52%
	Past-experience-based	139.24	142.78	142.35	140.98	145.4	142.15	2.28	27.01%
	Similar-taste-based	140.71	141.07	143.87	140.91	140.99	141.51	1.32	27.59%
	HTE by ORF	142.88	144.75	149.43	146.22	146.12	145.88	2.40	23.76%
	Multi-armed bandit	147.42	146.11	151.34	147.71	148.17	148.15	1.94	21.87%
Various exploration-exploitation heuristic	Greedy	161.21	160.72	163.18	165.21	164.09	162.88	1.89	10.84%
	$\epsilon - greedy$	163.08	164.47	166.71	164.31	164.07	164.52	1.33	9.74%
	Decaying $\epsilon - greedy$	167.72	165.42	170.32	167.12	168.93	167.90	1.85	7.53%
	VDBE	168.2	170.21	171.42	169.78	170.83	170.08	1.22	6.15%
	Boltzmann	165.08	164.87	165.01	169.71	167.71	166.47	2.15	8.45%
	Decaying Boltzmann	168.72	165.75	169.36	167.14	168.42	167.87	1.43	7.55%
	Boltzmann-Gumbel	170.85	173.08	174.47	175.41	175.59	173.88	1.96	3.83%
	Count-based exploration	172.34	174.84	177.02	175.84	177.62	175.53	2.08	2.86%
	Quantization-based uncertain learning	177.48	183.25	181.87	179.24	180.93	180.55	2.25	-
DQN variants	Noisy net	168.08	165.78	163.71	166.07	165.04	165.73	1.59	8.94%
	Prioritized replay	171.04	169.87	173.19	171.07	175.91	172.21	2.38	4.84%

Table 2 Simulated Online Testing – DRL and Baselines Model Comparison with Top-5 Simulators (Individual Simulator Results)

H. The adaptations and differences between the proposed heuristics and Tang et al. (2017)

The adaptations and differences are three-fold:

1) **Incentive rule**: two approaches are mainly different in bonus incentive rule (i.e., how the bonus is used). Tang et al. (2017) adopted an indirect incentive rule. The bonus was used as additional reward beyond the original immediate reward. The model parameter was updated using the newly constructed total rewards $(r + r^+)$. Thus, this approach indirectly drove the agent's decision by updating the parameters. However, such an indirect rule might have delayed effect, since the DRL algorithm usually adopted the experience replay strategy. For example, they updated the model parameters using the stored samples in a replay pool rather than the current samples at the current time step. Thus, it might take rounds and rounds to update the

model parameters to let the bonus take effect. Building on this, we adapted to a direct incentive rule to mitigate such a delay effect. We used bonus in objective function . In this way the bonus can directly affect the agent's action selection at each time step. Intuitively our proposed extension can explore faster than Tang et al. (2017).

2) **Bonus formula**: Tang et al. (2017) constructed bonus as inverse weight of state counts $r^+(s) = \beta/n(\psi(s))$. This cannot be used in our proposed direct incentive rule, since the bonus was only a function of state. If we add such a bonus to $Q(s,a)$, the bonus cannot affect the action recommendation since the bonus was not a function of action. Thus, we adapt the bonus as inverse weight of state-action pairs counts $B(s,a) = \beta/n(\psi(s),a)$. The bonus can be added to the $Q(s,a)$ to directly affect agent's decision at each time step.

3) **DRL algorithm**: Tang et al. (2017) used the proposed heuristics with Trust Region Policy Optimization (TRPO (Schulman et al. 2015)). TRPO belongs to the category of policy-based methods. We also tried to use the proposed heuristics on DQN which belongs to the value-based RL methods. There are two categories DRL approaches in general including value-based methods which estimate the optimal action-value function (e.g., Deep Q-Network, Sarsa) and policy-based methods which search directly for the optimal policy (e.g., REINFORCE, TRPO). Different DRL category algorithms might fit different heuristics. We also contributed to testing if such a heuristic works on value-based methods.

I. Add the relative days into modeling

In the main results, we use relative days as a study time unit. Each relative day represented the day that consumer had activities. There were no consumer activities (states) in the duration between "relative days". The main goal of the current study is to showcase the DRL can learn optimal sequential targeting policy. The simulator part is an auxiliary and secondary role. In the ideal case, DRL can learn from interacting with real users through exploration-exploitation. Due to the unavailability of real environment, we tackle the challenges by learning a simulator from historical data. So the LSTM is served to mimic the real consumers behaviors as close as possible. Although current LSTM/GRU/RNN did not consider the gaps between relative days, we show the superior power of DRL compared to baselines using multiple simulators and error bars.

To further verify if incorporating time decay will improve the prediction power of the LSTM for mimicking the consumer behaviors or not, we have now added an additional experiment here. In addition to the original 14-dimentional state features, we have added time gap between two active days as additional feature in the state variable. For day 1, the time gap is 0. For day 2, the time gap is the difference between previous activity date and this activity date. For a fair comparison, we now compare the reward prediction and original 14-d state feature prediction performance. We have adopted the multi-task LSTM and two single LSTM for a comparison as these are the top-2 performing model. The results are shown in following Table 3. We found that with relative days and without relative days models have very similar out-of-sample test results. This is true for both multi-task LSTM and two single LSTM. This indicates that adding relative days does not seem to significantly change the performance of simulators.

Algorithms	Relative or not	In-sample		Out-of-sample	
		Sate MSE	Reward MSE	Sate MSE	Reward MSE
Multi-task LSTM	Without relative days	0.51	0.41	0.58	0.47
	With relative days	0.52	0.43	0.56	0.48
Single-task LSTM	Without relative days	0.58	0.45	0.64	0.51
	With relative days	0.60	0.43	0.67	0.49

Table 3 Comparison of simulator performance: with and without relative days.

J. Selected Alpha for Multi-task Learning

We report the selected alpha for three multi-task learning models in Table 4. Originally, we did not separate two tasks. We combined the learning of the next state and reward i.e., (s, r) together and treated them as a single learning task. Because the state is 14-dimensional and reward is 1-dimensional, the MSE will bias toward state and prioritize the state learning. So we conducted multi-task learning to solve this issue. There were two prediction tasks - next state prediction and reward prediction. We set α as a hyperparameter representing the weight towards learning the next state variable (as opposed to the reward variable). Thus, the total loss of Multi-task learning was as follows:

$$\text{Total loss} = \alpha * \text{MSE}_{\text{Next state prediction}} + (1-\alpha) * \text{MSE}_{\text{Reward prediction}} \tag{26}$$

We experimented$\alpha = \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$ to balance two tasks and search for the optimal weight. We found out that the alphas for the next steps prediction are 0.3 or 0.4. This is highly aligned with our expectation. It assigned the next state prediction task less weight than reward prediction. In this way, it can solve our previous concern that the learning will bias toward the state.

Modesl	Selected alpha
Multi-task LSTM	0.4
Multi-task GRU	0.3
Multi-task RNN	0.4
Multi-task MLP	0.3

Table 4 Selected Alpha for Multi-task Learning

References

Bellman R (2013) Dynamic programming, courier corporation. *New York, NY* 707.

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 43–52 (Morgan Kaufmann Publishers Inc.).

Cesa-Bianchi N, Gentile C, Lugosi G, Neu G (2017) Boltzmann exploration done right. *Advances in neural information processing systems*, 6284–6293.

Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Friedman JH (2001) Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232.

Fudenberg D, Villas-Boas JM (2006) Behavior-based price discrimination and customer recognition. *Handbook on economics and information systems* 1:377–436.

Hauser JR, Urban GL, Liberali G, Braun M (2009) Website morphing. *Marketing Science* 28(2):202–223.

Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 30(1):76–80.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Mooney RJ, Roy L (2000) Content-based book recommending using learning for text categorization. *Proceedings of the fifth ACM conference on Digital libraries*, 195–204 (ACM).

Oprescu M, Syrgkanis V, Wu ZS (2018) Orthogonal random forest for causal inference. *arXiv preprint arXiv:1806.03467* .

Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. *International conference on machine learning*, 1889–1897 (PMLR).

Schwartz EM, Bradlow ET, Fader PS (2017) Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science* 36(4):500–522.

Tang H, Houthooft R, Foote D, Stooke A, Chen OX, Duan Y, Schulman J, DeTurck F, Abbeel P (2017) # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 2753–2762.

Tokic M (2010) Adaptive ε-greedy exploration in reinforcement learning based on value differences. *Annual Conference on Artificial Intelligence*, 203–210 (Springer).

Urban GL, Liberali G, MacDonald E, Bordley R, Hauser JR (2014) Morphing banner advertising. *Marketing Science* 33(1):27–46.

Wu H, Guo X, Liu X (2018) Adaptive exploration-exploitation tradeoff for opportunistic bandits. *International Conference on Machine Learning*, 5306–5314 (PMLR).

Zhang Y, Li B, Luo X, Wang X (2019) Personalized mobile targeting with user engagement stages: Combining a structural hidden markov model and field experiment. *Information Systems Research* .