Virtual-Taobao: Virtualizing Real-World Online Retail Environment for Reinforcement Learning*

Jing-Cheng Shi,[1,2] **Yang Yu,**[1] **Qing Da,**[2] **Shi-Yong Chen,**[1] **An-Xiang Zeng**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{shijc, yuy, chensy}@lamda.nju.edu.cn
[2]Alibaba Group
{jingcheng.sjc, daqing.dq}@alibaba-inc.com, renzhong@taobao.com

Abstract

Applying reinforcement learning in physical-world tasks is extremely challenging. It is commonly infeasible to sample a large number of trials, as required by current reinforcement learning methods, in a physical environment. This paper reports our project on using reinforcement learning for better commodity search in Taobao, one of the largest online retail platforms and meanwhile a physical environment with a high sampling cost. Instead of training reinforcement learning in Taobao directly, we present our environment-building approach: we build *Virtual-Taobao*, a simulator learned from historical customer behavior data, and then we train policies in Virtual-Taobao with no physical sampling costs. To improve the simulation precision, we propose GAN-SD (GAN for Simulating Distributions) for customer feature generation with better matched distribution; we propose MAIL (Multi-agent Adversarial Imitation Learning) for generating better generalizable customer actions. To further avoid overfitting the imperfection of the simulator, we propose ANC (Action Norm Constraint) strategy to regularize the policy model. In experiments, Virtual-Taobao is trained from hundreds of millions of real Taobao customers' records. Compared with the real Taobao, Virtual-Taobao faithfully recovers important properties of the real environment. We further show that the policies trained purely in Virtual-Taobao, which has zero physical sampling cost, can have significantly superior real-world performance to the traditional supervised approaches, through online A/B tests. We hope this work may shed some light on applying reinforcement learning in complex physical environments.

Introduction

With the incorporation of deep neural networks, Reinforcement Learning (RL) has achieved significant progress recently, yielding lots of successes in games (Mnih et al. 2013; Silver et al. 2016; Mnih et al. 2016), robotics (Kretzschmar et al. 2016), natural language processing (Su et al. 2016), etc. However, there are few studies on the application of RL in physical-world tasks like large online systems interacting with customers, which may have great influence on the user experience as well as the social wealth.

Large online systems, though rarely incorporated with RL methods, can indeed yearn for the embrace of RL (Hu et al. 2018). In fact, a variety of online systems involve the sequential decision making as well as the delayed feedbacks. For instance, an automated trading system needs to manage the portfolio according to the historical metrics and all the related information with a high frequency, and carefully adjusts its strategy through analyzing the long-term gains. Similarly, an E-commerce search engine observes a buyer's request, and displays a page of ranked commodities to the buyer, then updates its decision model after obtaining the user's feedback to pursue revenue maximization. During a session, it keeps displaying new pages according to latest information of the buyer if he/she continues to browse. Previous solutions are mostly based on supervised learning. They are incapable of learning sequential decisions and maximizing long-term reward. Thus RL solutions are highly appealing.

One major barrier to directly applying RL in these scenarios is that, current RL algorithms commonly require a large amount of interactions with the environment, which take high physical costs, such as real money, time from days to months, bad user experience, and even lives in medical tasks. To avoid physical costs, simulators are often employed for RL training. Google's application of data center cooling (Gao and Jamidar 2014) demonstrates a good practice: the system dynamics are approximated by a neural network, and a policy is later trained in the simulated environment via some state-of-the-art RL algorithms. In our project for commodity search in Taobao.com, we have the similar process: build a simulator, i.e, Virtual-Taobao, then the policy can be trained offline in the simulator by any RL algorithm maximizing long-term reward. Ideally, the obtained policy would perform equally well in the real environment, or at least provides a good initialization for cheaper online tuning.

However, different from approximating the dynamics of data centers, simulating the behavior of hundreds of millions of customers in a dynamic environment is much more challenging. We treat customers behavior data to be generated from customers' policies. Deriving a policy from data can be realized by existing *imitation learning* approaches (Schaal 1999; Argall et al. 2009). The behavior cloning (BC) methods (Pomerleau 1992) learn a policy mainly by supervised

*This work is supported by the National Key R&D Program of China (2018YFB1004300), NSFC (61876077), Jiangsu SF (BK20170013), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Yang Yu is the corresponding author. This work is done when Jing-Cheng Shi was an intern in Alibaba Group, whom is now affiliated with Alibaba Group.

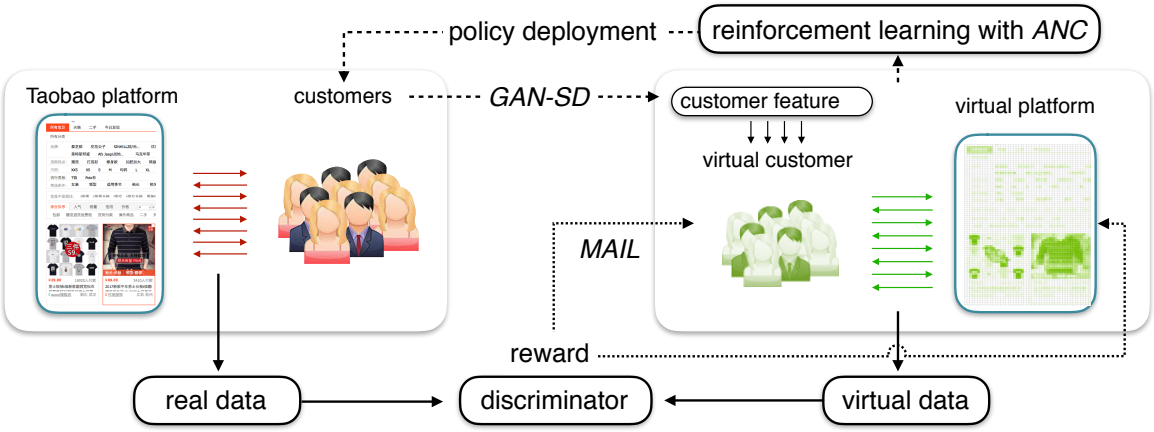


Figure 1: The Virtual-Taobao architecture. First, GAN-SD learns to generate customers features. With customer features, the behavior of the customer is then learned by the MAIL approach. After training the Virtual-Taobao environment, the platform policy is then trained in Virtual-Taobao with ANC. Finally, the platform policy is directly deployed in the real Taobao.

methods from the state-action data. BC requires the i.i.d. assumption on the demonstration data that is unsatisfied in RL tasks. The inverse reinforcement learning (IRL) methods (Ng, Russell, and others 2000) learn a reward function from the data, and a policy is then trained according to the reward function. IRL relaxes the i.i.d. assumption of the data, but still assumes that the environment is static. When the environment, i.e. Taobao platform, is changing, the learned policy could fail. All the above issues make these method less practical for building the Virtual-Taobao.

In this work, we build Virtual-Taobao through generating customers and generating interactions. Customers with search request, which have a very complex and widely spanned distribution, come into Taobao and trigger the platform search engine. However, sampling from the database can not generate out-of-data customers, which results in low generalization of the final model. We propose the GAN-for-Simulating-Distribution (GAN-SD) approach to generate virtual customers, as we find traditional approaches, such as GMM and GANs, do not work well for such high-dimensional data. To generate interactions, which is the key component of Virtual-Taobao, we propose the Multi-agent Adversarial Imitation Learning (MAIL) approach. We could have directly invoked Taobao platform strategy in Virtual-Taobao, which however makes a static environment that will not adapt to changes in the real environment. Therefore, MAIL learns the customers' policies and the platform policy simultaneously. In order to learn the two sides, MAIL follows the idea of GAIL (Ho and Ermon 2016) using generative adversarial framework (Goodfellow et al. 2014). MAIL trains a discriminator to distinguish the simulated interactions from the real interactions; the discrimination signal feeds back as the reward to train the customer and platform policies for generating more real-alike interactions. After generating customers and interactions, Virtual-Taobao has been accomplished, and is used to train the platform policy. However, we notice that reinforcement learning algorithm can be powerful enough to overfit the imperfection of the Virtual-Taobao, which implies that it can do quite well in the virtual environment but poorly in the real. We thus propose Action Norm Constraint (ANC) strategy to regularize the policy. In the experiments, we build Virtual-Taobao from hundreds of millions of customer records, and compare it with the real environment. Our results disclose that Virtual-Taobao successfully reconstructs properties very close to the real environment. We then employ Virtual-Taobao to train platform policy for maximizing the revenue. Comparing with the traditional supervised learning approach, the strategy trained in Virtual-Taobao achieves more than 2% improvement of revenue in the real environment, with zero cost of physical trials.

The rest of the sections present the background, the Virtual-Taobao approach, the offline and online experiments, and the conclusion, in order.

Background

Reinforcement Learning

Reinforcement learning (RL) solves sequential decision making problems through trial-and-error. We consider a standard RL formulation based on a Markov Decision Process (MDP). An MDP is described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma \rangle$ where $\mathcal{S}$ is the observation space, $\mathcal{A}$ is the action space and γ is the discount factor. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function to generate next states from the current state-action pair. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function. At each timestep t, the RL agent observes a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$ following a policy π. Then it will be transfered into a new state s_{t+1} according to $\mathcal{T}(s_t, a_t)$ and receives an immediate reward $r_t = \mathcal{R}(s_t, a_t)$. The reward and the next state depend on the current state and action, but are independent of all the previous states and actions (Markov property). $R_t = \sum_{k=0}^{\infty} \gamma^t r_{t+k}$ is the discounted, accumulated reward with the discount factor γ. The value function $V^\pi(s_t) = E[R_t | s_t = s]$ is an ap-

proximation of the expected R_t, measuring how good each state is, following the policy π. The action-value function $Q^\pi(s_t, a_t) = E[R_t \,|\, s_t = s, a_t = a]$ measures how good each state-action pair is. The advantage function is defined as $A_t^\pi = A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$. The goal of RL is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ that solves the MDP by maximizing the expected discounted return, i.e., $\pi^* = \arg\max E[\, R_t \,]$

Trust Region Policy Optimization (Schulman et al. 2015) which we employ in this work is one of the state-of-the-art policy gradient methods, which solves the problem with a policy shifting constraint. Specifically,

$$\begin{aligned} &\text{maximize}_\theta E_t\Big[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}} A_t\Big] \\ &\text{subject to } \; E_t[\text{KL}[\pi_{\theta_{old}}, \pi_\theta]] \leq \delta \end{aligned} \tag{1}$$

Here, $\pi_{\theta_{old}}$ is the policy before the update. The Eq.(1) can be approximately solved by the conjugate gradient algorithm, after making a linear approximation to the objective and a quadratic approximation to the constraint.

Imitation Learning

Rather than learning a policy from scratch as in RL, it is apparently a more effective and a natural task to act following by an experienced teacher. What's more, in traditional RL, the manual-designed reward function may not be the real one (Farley and Taylor 1991; Hoyt and Taylor 1981) and a tiny change of reward function in RL may result in a totally different policy. In imitation learning, the agent is given trajectory samples from some expert policy, and infers the expert policy.

Behavior Cloning & Inverse Reinforcement Learning are two traditional approaches for imitation learning: behavior cloning learns a policy as a supervised learning problem over state-action pairs from expert trajectories (Pomerleau 1992); and inverse reinforcement learning finds a reward function under which the expert is uniquely optimal (Russell 1998), and then train a policy according to the reward.

While behavior cloning methods are powerful for one-shot imitation (Duan et al. 2017), it needs large training data to work even on non-trivial tasks, due to compounding error caused by covariate shift (Ross and Bagnell 2010). It also tends to be brittle and fails when the agent diverges too much from the demonstration trajectories (Ross, Gordon, and Bagnell 2011). On the other hand, inverse reinforcement learning finds the reward function being optimized by the expert, so compounding error, a problem for methods that fit single-step decisions, is not an issue. IRL algorithms are often expensive to run because they need reinforcement learning in a inner loop. Some works have focused on scaling IRL to large environments (Finn, Levine, and Abbeel 2016).

Generative Adversarial Networks

Generative adversarial networks (GANs) (Goodfellow et al. 2014) and its variants are rapidly emerging unsupervised machine learning techniques. They are implemented by a system of two neural networks contesting with each other in a zero-sum game framework. They train a discriminator D to maximize the probability of assigning the correct labels to both training examples and generated samples, and a generator G to minimize the classification accuracy according to D. The discriminator and the generator are implemented by neural networks, and are updated alternately in a competitive way. Recent studies have shown that GANs are capable of generating faithful real-world images (Zhu et al. 2017), implying their applicability in modeling complex distributions.

Generative Adversarial Imitation Learning (Ho and Ermon 2016) was recently proposed to overcome the brittleness of behavior cloning as well as the expensiveness of inverse reinforcement learning using GAN framework. GAIL allows the agent to interact with the environment and learns the policy by RL methods while the reward function is improved during training. Thus the RL method is the generator in the GAN framework. GAIL employs a discriminator D to measure the similarity between the policy-generated trajectories and the expert trajectories. GAIL is proven to achieve the similar theoretical and empirical results as IRL and GAIL is more efficient. GAIL has become a popular choice for imitation learning (Kuefler et al. 2017) and there already exist model-based (Baram, Anschel, and Mannor 2016) and third-person (Stadie, Abbeel, and Sutskever 2017) extensions.

Virtual-Taobao

Problem Description

Commodity search is the core business in Taobao, one of the largest retail platforms in China. Taobao can be considered as a system where the search engine interacts with customers. The search engine in Taobao deals with millisecond-level responses to billions of commodities, while the customers' preference of commodities are also rich and diverse. From the engine's point of view, Taobao platform works as the following. A customer comes and sends a search request to the search engine. Then the engine makes an appropriate response to the request by sorting the related commodities and displaying the page view (PV) to the customer. The customer gives the feedback signal, e.g. buy something, turn to the next page, or leave Taobao, according to the PVs as well as the buyer's own intention. The search engine receives the feedback signal and makes a new decision for the next PV request. One of the business goals in Taobao is to maximize sales by optimizing the strategy of displaying PVs.

As the feedback signal, i.e, the customer behavior, influenced by previous PVs, it's more reasonable to consider it as a multi-step decision problem rather than a one-step supervised learning problem while optimizing the search engine policy. Therefore, considering the search engine as an agent, and the customers' feedback as the corresponding environment, the commodity search in Taobao is a sequential decision making problem. It is reasonable to assume customers can only remember a limited number, m, of the latest PVs, which means the feedback signals are only influenced by m historical actions of the search agent. Let $a \in \mathcal{A}$ denote the action of the search engine agent and F_a denote the feedback distribution from customers at a , given the historical actions from the engine $a_0, a_1, ..., a_{n-1}$, we have

$$F_{a_n|a_{n-1},a_{n-2},...,a_0} = F_{a_n|a_{n-1},...,a_{n-m}} \tag{2}$$

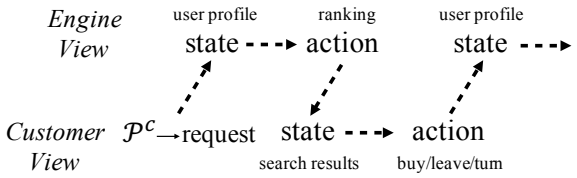


Figure 2: Taobao search in engine view and in customer view

Note that if we assume $m = 1$, i.e. the customers' feedback is only influenced by the last engine action, this implies a standard Markov Decision Process.

On the other hand, if we consider the customer as an agent and the search engine as the environment, the shopping process for a customer can also be viewed as a sequential decision process. The customer response to the ranked commodities, i.e. the action of the search engine. As a customer's behavior, i.e. the feedback signals, is influenced by the last m PVs, which are generated by the search engine and are affected by the last feedback from the customer. The customers' behaviors also have the Markov property. The process of developing the shopping policy for a customer can be regarded as a process optimizing his shopping preference in Taobao.

Figure 2 shows the engine and customers are the environments of each other and their policies are coupled together. Specifically, we are going to make some notes as follows. We use $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \pi \rangle$ and $\mathcal{M}^c = \langle \mathcal{S}^c, \mathcal{A}^c, \mathcal{T}^c, \mathcal{R}^c, \pi^c \rangle$ to represent the decision process for engine and customers. Note that the customers with requests come from the real environment $\mathcal{P}^c$.

The observation of the engine, which is designed to contain the customer feature and search request, remains the same if the customer just turns to the next page without other behavior. The state changes if the customer sends another request or leaves Taobao.

Comparing to the search engine, the customer individual is more sensitive to the environment, so we make some specific design for the customer. The customer behavior will be influenced by what he/she wants and what he/she sees, which will be reflected by $\mathcal{S}$ and $\mathcal{A}$, in which $\mathcal{S}$ is the engine observation, i.e. customer feature with request, and $\mathcal{A}$ is the engine action, i.e. the displayed page view. Considering customer's purchase intention changes with the number of pages, we let $\mathcal{S}^c = \mathcal{S} \times \mathcal{A} \times \mathbb{N}$, where $\mathbb{N}$ denotes the page index space. Note that $s^c \in \mathcal{S}^c$ can be written as $\langle s, a, n \rangle \in \mathcal{S} \times \mathcal{A} \times \mathbb{N}$.

The trasition functions are defined as follows:

$$\mathcal{T}(s,a) = \begin{cases} s, & \text{if } a^c = \text{turn page} \\ s' \sim \mathcal{P}^c, & \text{otherwise} \end{cases}$$

$$\mathcal{T}^c(s^c,a^c) = \begin{cases} \langle s' \sim \mathcal{P}^c, \pi(s'), 0 \rangle, & \text{if } a^c = \text{leave} \\ \langle s, a, n+1 \rangle, & \text{if } a^c = \text{turn page} \\ \text{terminates}, & \text{if } a^c = \text{buy, or } n > \text{MaxIndex} \end{cases}$$

For the engine, if the customer buys something, we give the engine a reward of 1 otherwise 0. For customers, the reward function is currently unknown to us.

GAN-SD: Generating Customer Features

To build the Virtual-Taobao, we need to firstly generate the customer features, i.e. sample a user U^c that includes its request from $\mathcal{P}^c$ to trigger the interaction process. The generated distribution of the customers should have a similar distribution with the real one.

Learning a distribution in a high-dimensional space is challenging. Classical approaches such as GMM (Gaussian Mixture Model) is hard to approximate the distribution well. It's known that Generative Adversarial frameworks are designed to generate samples close to the original data and achieves great success for generating images. However, as the traditional GANs discriminators decide whether instances comes from the real, they lack the ability to capture the distribution architecture of customers. To generate a distribution rather than a single instance, we propose Generative Adversarial Networks for Simulating Distribution (GAN-SD), as in Algorithm 1. Similar to GANs, GAN-SD maintains a generator G and a discriminator D. The discriminator tries to correctly distinguish the generated data from the training data by maximizing the following objective function

$$E_{p_{x\sim\mathcal{D}}}[\log D(x)] + E_{p_{z\sim\mathcal{G}}}[\log(1 - D(G(z)))],$$

while the generator is updated to maximize the following objective function

$$E_{p_{\mathcal{D}}, p_{\mathcal{G}}}[D(G(z)) + \alpha\mathcal{H}(V(G(z))) - \beta KL(V(x)||V(G(z)))].$$

Here we shorten $p_{x\sim\mathcal{D}}, p_{z\sim\mathcal{G}}$ to $p_{\mathcal{D}}, p_{\mathcal{G}}$ for space limitation. $G(z)$ is the generated instance from the noise sample z. $V(\cdot)$ denotes some variable associated with the inner value. In our implementation, $V(\cdot)$ is the customer type of the instance. $\mathcal{H}(V(G(z)))$ denotes the entropy of the variable from the generated data, which is used to make a wider distribution. $KL(V(x)||V(G(z)))$ is the KL divergence between the variables from the training data to the generated data, which is used to guide the generated distribution by the distribution in the training data. With the KL divergence and entropy constraints, GAN-SD learns a generator with more guided information form the real data, and can generate much better distribution than the traditional GANs.

MAIL: Generating Interactions

We then generate interactions between the customers and the platform, which is the key of Virtual-Taobao, through simulating customer policy. We accomplish this by proposing the Multi-agent Adversarial Imitation Learning (MAIL) approach following the idea of GAIL. GAIL allows the agent interact with the environment during training while the reward function keeps being optimized. Note that the environment should be accessible during GAIL training. However, training customer policies needs the engine as the environment which is unknown or dynamics.

Different from GAIL that trains one agent policy in a static environment, MAIL is a multi-agent approach that trains the customer policy as well as the engine policy. In such way, the learned customer policy is able to generalize with different engine policies. As MAIL trains the two policies together, i.e. the agent and the environment, it only needs the historical data and do not need to access the real environment.

Algorithm 1 GAN-SD

1: **Input:** Real data distribution $\mathcal{P}_{\mathcal{D}}$
2: Initialize training variables θ_D, θ_G
3: **for** $i = 0, 1, 2...$ **do**
4: **for** k steps **do**
5: Sample minibatch from $p_{\mathcal{G}}$
6: Sample minibatch from $p_{\mathcal{D}}$
7: Update the generator by gradient:

$$\nabla_{\theta_G} E_{p_{\mathcal{G}}, p_{\mathcal{D}}}[D(G(z)) + \alpha\mathcal{H}(G(z)) - \beta KL(x||G(z))]$$

8: **end for**
9: Sample minibatch from $p_{\mathcal{G}}$
10: Sample minibatch from $p_{\mathcal{D}}$
11: Update the discriminator by gradient:

$$\nabla_{\theta_D} E_{p_{x\sim\mathcal{D}}}[\log D(x)] + E_{p_{x\sim\mathcal{G}}}[\log(1 - D(G(z)))]$$

12: **end for**
13: **Output:** Customer generator G

Algorithm 2 MAIL

1: **Input:** Expert trajectories τ_e, customer distribution $\mathcal{P}^c$
2: Initialize variables κ, σ, θ
3: **for** $i = 0, 1, 2..., I$ **do**
4: **for** $j = 0, 1, 2..., J$ **do**
5: $\tau_j = \emptyset$, $s \sim \mathcal{P}^c, a \sim \pi_\sigma(s, \cdot), s^c = \langle s, a \rangle$
6: **while** NOT TERMINATED **do**
7: sample $a^c \sim \pi(s^c, \cdot)$,
add (s^c, a^c) to τ_j,
generate $s^c \sim \mathcal{T}^c_\sigma(s^c, a^c|\mathcal{P}^c)$
8: **end while**
9: **end for**
10: Sample trajectories τ_g from $\tau_{0\sim J}$
11: Update θ to in the direction to minimize

$$E_{\tau_g}[\log(\mathcal{R}^c_\theta(s, a))] + E_{\tau_e}[\log(1 - \mathcal{R}^c_\theta(s, a))]$$

12: Update κ, σ by optimizing $\pi^c_{\kappa,\sigma}$ with RL in $\mathcal{M}^c$
13: **end for**
14: **Output:** The customer agent policy π^c

However, learning the agent, i.e. the customer policy, and the environment, i.e. the engine policy, iteratively, could results in a very large search space, and thus poor performance. Fortunately, we can optimize them jointly. In MAIL, to imitate the customer agent policy π^c, we parameterize the customer policy π^c_κ by κ, the search engine policy π_σ by σ, and the reward function $\mathcal{R}^c_\theta$ by θ. We also denote $\mathcal{T}^c$ as $\mathcal{T}^c_\sigma$. As the observation of customers $s^c = \langle s, a, n \rangle$ depends on the engine action a, we have:

$$\pi^c(s^c, a^c) = \pi^c(\langle s, a, n \rangle, a^c) = \pi^c(\langle s, \pi(s, \cdot), n \rangle, a^c)$$

which indicates that, given the engine policy, the joint policy $\pi^c_{\kappa,\sigma}$ can be viewed as a mapping from $\mathcal{S}\times\mathbb{N}$ to $\mathcal{A}^c$. In another word, given the parameters of the search agent, the customer agent can directly make his decision without PV information. As $\mathcal{S}^c = \mathcal{S} \times \mathcal{A} \times \mathbb{N}$, we still consider $\pi^c_{\kappa,\sigma}$ as a mapping from $\mathcal{S}^c$ to $\mathcal{A}^c$ for convenience. The formalization of joint policy brings the chance to simulate π and π^c together, i.e., learning the agent policy and the environment together. The reward function $\mathcal{R}^c$ is designed as the non-discriminativeness of the generated data and the historical state-action pairs. A employed RL algorithm will maximize the reward, implying generating indistinguishable data.

Algorithm 2 displays the procedure of MAIL. To run MAIL, we need the historical trajectories τ_e and a customer distribution $\mathcal{P}^c$, which is required by $\mathcal{T}^c_\sigma$. In this paper, $\mathcal{P}^c$ is learned by GAN-SD in advance. After initializing the variables, we start the main procedure of MAIL: In each iteration, we collect trajectories during the interaction between the customer agent and the environment (line 4-9). Then we sample from the generated trajectories and optimize reward function by a gradient method (line 10-11). Then, κ, τ will be updated by optimizing the joint policy $\pi^c_{\kappa,\sigma}$ in $\mathcal{M}^c$ with a RL method (line 12). When the iteration terminates, MAIL returns the customer agent policy π^c.

After simulating the $\mathcal{P}^c$ and π^c, which means we know how customers are generated and how they response to PVs, Virtual-Taobao has been built. We can generate the interactions by deploying the engine policy to the virtual environment.

ANC: Reduce Overfitting to Virtual-Taobao

We observe that if the deployed policy is similar to the historical policy, Virtual-Taobao tends to behave more similarly than if the two policies are completely different. However, a similar policy means little improvement. A powerful RL algorithm, such as TRPO, can easily train an agent to overfit Virtual-Taobao, which means it can perform well in the virtual environment but poorly in the real environment. Actually, we need to trade off between the accuracy, i.e. similar to the historical policies and the improvement, i.e., away from the historical policies. However, as the historical engine policy is unavailable in practice, we can not measure the distance between a current policy and the historical one. Fortunately, we note that the norms of most historical actions are relatively small, we propose the Action Norm Constraint (ANC) strategy to control the norm of taken actions. That is, when the norm of the taken action is larger than the norms of most historical actions, we give the agent a penalty. Specifically, we modified the original reward function as follow:

$$r'(s, a) = \frac{r(s, a)}{1 + \rho \max\{\|a\| - \mu, 0\}} \tag{3}$$

Empirical Study

Experiment Setting

To verify the effect of Virtual-Taobao, we use following measurements as indicators.

- **Total Turnover (TT)** The value of the commodities sold.
- **Total Volume (TV)** The amount of the commodities sold.
- **Rate of Purchase Page (R2P)** The ratio of the number of PVs where purchase takes place.

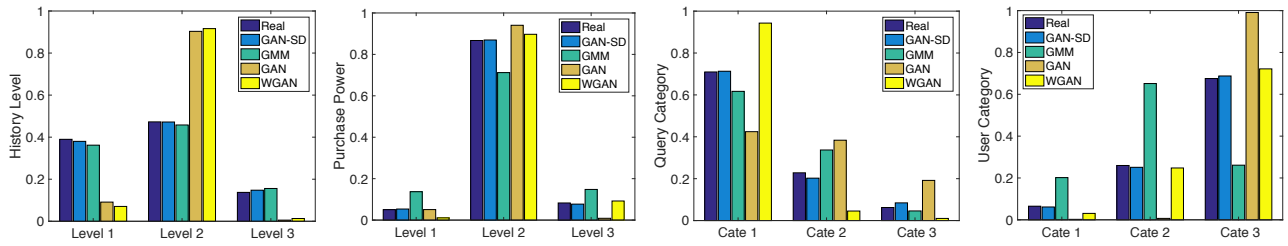


Figure 3: Comparisons of the learned customer distributions.

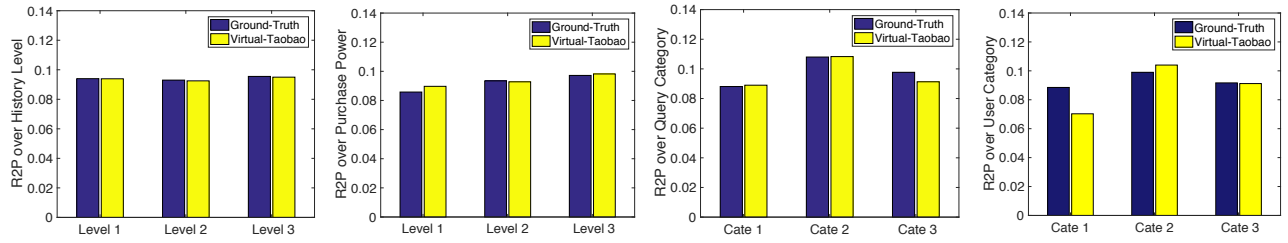


Figure 4: Comparisons of R2P between the real Taobao and Virtual-Taobao.

All the measurements are adopted in online experiments. In offline experiments we only use R2P as we didn't predict the customer number and commodity price in this work. For the convenience of comparing these indicators between the real and virtual environment, we deployed a random engine policy in the real environments (specifically, an online A/B test bucket in Taobao) in advance and collect the corresponding trajectories as the historical data (about 400 million records). Note that, we have no assumption of the engine policy which generates the data, i.e., it could be any unknown complex model while we build the virtual environment.

We simulate the customer distribution $\mathcal{P}^c$ by GAN-SD with $\alpha = \beta = 1$. Then we build Virtual-Taobao by MAIL. The RL method in MAIL we used is TRPO. All function approximators in this work are implemented by multi-layer perceptions. Due to the resource limitation, we can only compare two policies at the same time in the online experiments.

Firstly, we show that trough Algorithm 1 the virtual customers have a similar distribution of the real. We deploy the historical engine policy in Virtual-Taobao and compare the R2P over time and customer features of customers in the virtual environment and the real one. Then to show the effect of ANC strategy, we run an RL algorithm in the virtual environment with and without the constraints and compare the two result policies in the real environment. At last we run an RL algorithm in the virtual environment and run two supervised learning algorithms on the historical data, i.e., RL+VTaobao v.s. SL+Data. We deploy the result policies in the real environment for 24 hours and RL policy is better than the SL policies.

On Virtual-Taobao Properties

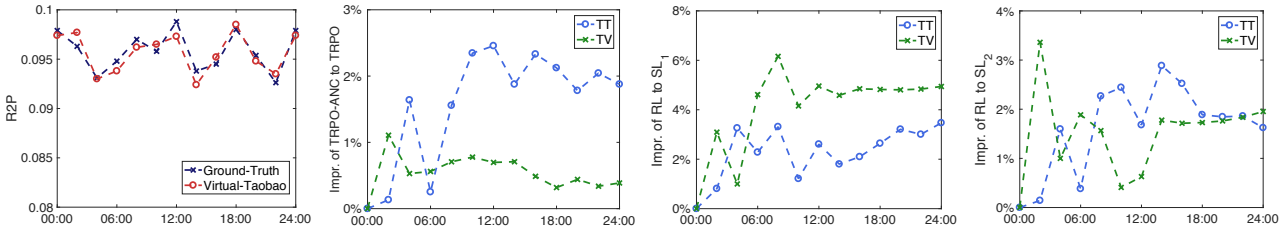
Virtual Customer Distributions and Behaviors

The proportion of different customers is a basic criterion for testing Virtual-Taobao. We simulate the customer distribution $\mathcal{P}^c$ by GAN-SD, and generate 100,000 customers by $\mathcal{P}^c$ and calculate the proportions over four dimensions respectively, i.e. the level of historical turnovers, the purchase power, the category of query words and the category of customers. We also simulate the distribution by GMM, GAN and WGAN(Arjovsky, Chintala, and Bottou 2017). We compare the result with the ground truth, i.e., distributions in Taobao. Figure 3 indicates that distribution by GAN-SD is the most similar to the ground truth and Table 1 displays the specific KL divergence from the generated customers to the real. In the experiment, the cluster number of GMM was carefully chosen to 10 to fit the data and we find traditional GANs hard to capture the distribution structure.

Then, we generate virtual customers through GAN-SD and build Virtual-Taobao through MAIL. People have different preference of consumption which reflects in R2P. To show whether Virtual-Taobao can simulate the real, we report the influence of customer features on the R2P in Virtual-Taobao and compare it with the results in the real. As shown in Figure 4, the results in Virtual-Taobao are quite similar with the ground truth.

R2P in Virtual-Taobao over Time

R2P of the customers varies with time in Taobao, thus Virtual-Taobao should have the similar property. Since our customer model is independent of time, we divide the one-day historical data into 12 parts in order of time to simulate process of R2P changing over time. We train a virtual environment on every divided dataset independently. Then we deploy the same historical engine policy, i.e. the random policy, in the virtual environments respectively. We report the R2Ps in the virtual and the real environment. Figure 5(a) indicates Virtual-Taobao can reflect the trend of R2P over time.



(a) R2P over time (b) Impr. by ANC (c) Impr. to SL_1 (d) Impr. to SL_2

Figure 5: (a) R2P over time. (b) Impr. by ANC. (c) Impr. to SL_1. (d) Impr. to SL_2.

Table 1: The KL divergence from virtual to real

KL	History	Purchase	Query	User
GAN-SD	**0.00**	**0.00**	**0.01**	**0.00**
GMM	0.03	0.03	0.07	0.32
GAN	0.70	0.11	0.18	0.90
WGAN	0.68	0.03	0.28	0.02

Table 2: The R2P improvement from random in two simulators by behavior clone and MAIL

	0 day	1 day	1 week	1 month
By BC	21.65%	8.33%	1.04%	-0.54%
By MAIL	18.56%	16.67%	8.34%	8.49%

Reinforcement Learning in Virtual-Taobao

Generalization Capability of ANC

Virtual-Taobao should have generalization ability as it is built from the past but serves for the future. We run the TRPO algorithm in Virtual-Taobao and it raises R2P up to 0.3, which is almost impossible in Taobao. It is obvious that the TRPO agent over fits to Virtual-Taobao environment. We then retrain the TRPO agent with the ANC strategy in which $\rho = 1$ and $\mu = 0.01$, and R2P is decreased to 0.115 in Virtual-Taobao which is more acceptable.

We compare the policies by TRPO and TRPO-ANC in the real Taobao, Figure 5(b) shows the TT and TV increase of TRPO-ANC to TRPO. The policy by TRPO-ANC is always better than TRPO which indicates that the ANC strategy can reduce overfitting to Virtual-Taobao and has better ability of generalizing to the real environment.

Generalization Capability of MAIL

Then, we'll verify whether the policy from MAIL has better capability of generalization. As there is no previous work about how to construct Virtual-Taobao, we compare MAIL with the traditional Behavior Clone(BC) algorithm, i.e., learning a mapping from customer states to actions. BC algorithms assumes the data is i.i.d which is not true in practice. Thus the BC algorithms can't get the real shopping intention of customers and little changes of the environment will bring a large decrease of accuracy. However, MAIL learns the long term behavior without the i.i.d assumption thus is expected to have a better ability of generalization.

We build a virtual environment by MAIL from one-day's data and build another 3 environments by MAIL whose data is from one day, a week and a month later. We run TRPO-ANC in the first environment and deploy the result policy in other environments. The R2Ps are expected to decrease because the environments become more different as time goes and this can reflect the ability of generalization to new environment. We repeat the same process except that we replace MAIL with a behavior cloning method (BC) which uses the same network structure with MAIL. Table 2 shows the R2P improvement to a random policy. The R2P drops faster in the BC environment. The policy in BC environment even performs worse than then random policy after a month.

Experiments in the real environment

As the target of building Virtual-Taobao is to train RL algorithms offline, we learns polices by RL methods on Virtual-Taobao (*RL+VTaobao*) and by supervised learning methods on the historical data (*SL+Data*), then deploy them in the real environment for 24 hours. Note that Virtual-Taobao is constructed only from the historical data.

To learn a engine policy using supervised learning method, i.e. a map from $\mathcal{S}$ to $\mathcal{A}$, we divide the historical data S into two parts: S_1 contains the records with purchase and S_0 contains the other.

The first benchmark method, denoted by SL_1, is a classic regression model which intuitively attemps to fit a model to generates actions close to the 'right ones'.

$$\pi^*_{SL_1} = \text{argmin}_\pi \frac{1}{|S_1|} \sum_{(s,a)\in S_1} |\pi(s) - a|^2 \tag{4}$$

The benchmark SL_1 only uses the data S_1. To make full use of the historical data, the second benchmark SL_2 modified the loss function and the optimal policy is defined as follow which means that we want the generated actions close to the 'right ones' as well as far away from the 'wrong ones'. In addition, we add a penalty term on the norm of actions $\pi(s)$ to avoid the second term $|\pi_\theta(s) - a|^2$ playing the leading role when the action is too large.

$$\begin{aligned}\pi^*_{SL_2} = \text{argmin}_\pi &\frac{1}{|S_1|} \sum_{(s,a)\in S_1} |\pi(s) - a|^2 \\ &- \frac{\lambda_1}{|S_2|} \sum_{(s,a)\in S_2} |\pi(s) - a|^2 + \frac{\lambda_2}{|S|} \sum_{(s,a)\in S} |\pi(s)|^2\end{aligned} \tag{5}$$

In our experiment, λ_1 and λ_2 were set to 0.3 and 0.001 respectively. As we can only compare two algorithms in the real environment in the same time due to the resource limitation, we report the results of RL v.s. SL_1 and RL v.s. SL_2 respectively. The results of TT and TV improvements to SL policies are shown in Figure 5(c) and Figure 5(d) . The R2P in the real environment of SL_1, SL_2 and RL are 0.096, 0.098 and 0.101 respectively. Note that it's not an easy thing to improve TT and TV by even 1%. The *RL+VTaobao* policy has better performance than *SL+Data* policies. This means that Virtual-Taobao is a feasible solution to apply reinforcement learning in Taobao platform.

Conclusion

To overcome the high cost of training reinforcement learning for commodity search in the physical world of Taobao, we build Virtual-Taobao simulator, which is trained from historical data. We firstly generate virtual customers by GAN-SD, and generate virtual interactions through MAIL. The empirical results have verified that Virtual-Taobao can reflect properties of the real environment faithfully. We then train better platform policies with proposed ANC strategy in the Virtual-Taobao, resulting in platform policy with better real environment performance than the traditional supervised learning approaches. Virtualizing Taobao is meaningful and challenging. We hope this work can shed some light for applying reinforcement learning to complex physical tasks.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5):469–483.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Baram, N.; Anschel, O.; and Mannor, S. 2016. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*.

Duan, Y.; Andrychowicz, M.; Stadie, B.; Ho, J.; Schneider, J.; Sutskever, I.; Abbeel, P.; and Zaremba, W. 2017. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*.

Farley, C. T., and Taylor, C. R. 1991. A mechanical trigger for the trot–gallop transition in horses. *Science* 253(5017):306–308.

Finn, C.; Levine, S.; and Abbeel, P. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML'16*.

Gao, J., and Jamidar, R. 2014. Machine learning applications for data center optimization. *Google White Paper*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS'14*.

Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *NIPS'16*.

Hoyt, D. F., and Taylor, C. R. 1981. Gait and the energetics of locomotion in horses. *Nature* 192(16):239–240.

Hu, Y.; Da, Q.; Zeng, A.; Yu, Y.; and Xu, Y. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD'17*.

Kretzschmar, H.; Spies, M.; Sprunk, C.; and Burgard, W. 2016. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research* 35(11):1289–1307.

Kuefler, A.; Morton, J.; Wheeler, T.; and Kochenderfer, M. 2017. Imitating driver behavior with generative adversarial networks. In *IV'17*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML'16*.

Ng, A. Y.; Russell, S.; et al. 2000. Algorithms for inverse reinforcement learning. In *ICML'00*.

Pomerleau, D. A. 1992. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation* 3(1):88–97.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. In *AISTATS'10*.

Ross, S.; Gordon, D.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS'11*.

Russell, S. 1998. Learning agents for uncertain environments. In *COLT'98*.

Schaal, S. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3(6):233–242.

Schulman, J.; Levine, S.; Abbeel, S.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *ICML'15*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Stadie, B. C.; Abbeel, P.; and Sutskever, I. 2017. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*.

Su, P.; Gasic, M.; Mrksic, N.; Rojas-Barahona, L.; Ultes, S.; Vandyke, D.; Wen, T.; and Young, S. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Zhu, J.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV'17*.