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

摘要

通过对虚拟淘宝论文的复现工作，我成功重新构建了基于强化学习的淘宝商品搜索引擎代理及其训练环境的系统，并取得了较为满意的结果。复现的目标是根据先前提出的虚拟淘宝环境构建方法，构建一个强化学习训练的虚拟环境，以有效解决在物理世界任务中应用强化学习时面临的挑战性问题，为云计算模拟环境的构建工作奠定基础。按照论文中描述的方法，我构建了虚拟淘宝环境，这是一个基于数亿真实淘宝客户行为数据学习而来的模拟器。通过这个虚拟环境，代理能够避免物理采样成本，直接进行有效的强化学习策略训练。根据论文提出的模型架构，我将复现过程大致分为三个阶段：首先复现 GAN-SD 算法，用于改进客户特征的生成，使其更好地匹配真实分布。接着通过复现 MAIL 方法验证对抗性学习对强化学习行为策略预测的有效性。最后，为了避免过度拟合虚拟淘宝环境的缺陷，我复现了 ANC 策略，用于规范策略模型，确保其在真实世界中具有良好的性能。通过上面三个阶段的工作，我最终构建了一个仿真的虚拟淘宝环境并进行验证性实验。在实验中，通过论文提供的在线 A/B 测试集展示了直接在虚拟淘宝中训练的策略相较于传统监督方法在真实世界中的显著优越性。该复现工作旨在验证论文方法在解决强化学习代理在物理环境学习成本过高问题上的有效性，为云计算资源缩放模拟环境的构建打下坚实基础。我相信这份复现报告将为进一步的研究和实践提供有益的参考，为解决现实世界问题提供更多可行的方案。

关键词：虚拟环境；强化学习；GAN-SD；MAIL

1 引言

为什么选择这篇论文进行复现？我当前的科研方向为强化学习在云计算资源管理上的应用。传统的云资源缩放策略（如基于规则、分析建模）采用的是分而治之的思想，对每一个服务设置阈值来进行资源的缩放。但是这种方式没有考虑多服务应用程序中的服务相关性。我们知道在传统的强化学习 (RL,Q-learning) 中，智能体 (agent) 通过与环境进行交互来学习最佳策略。它通过尝试不同的行动并根据奖励信号来调整策略，因此强化学习被常常用来制定整体最大收益的序列策略。在导师的建议下我已经尝试利用强化学习代理 Agent 替换原有的 kubernetes 资源缩放机制 HPA 来进行资源的监控和缩放，在保证服务质量的情况下成功地实现了长期收益最大化。但是资源的缩放意味着负载容器的销毁、重建，缩放的过程不但等待

时间长且耗能大。也就是说考虑到物理环境训练成本高，在真实的集群环境中进行强化学习策略训练是不切实际的。如何构建一个虚拟环境实现强化学习代理的预训练是十分值得探讨的问题。而虚拟淘宝这篇论文探究了如何构建一个虚拟环境用于对基于强化学习的淘宝商品搜索引擎代理进行策略训练。根据虚拟淘宝系统架构，我将复现工作分为三个部分，并利用训练好的模型与监督学习进行了对比性实验，结果表明利用构建的虚拟环境训练的强化学习代理不但很快实现收敛同时还能实现更好的推荐。我希望借助复现这篇论文学习虚拟环境构建过程中可能面临的问题与挑战、创新性思路、相关算法的可行性和有效性，为后期云资源虚拟环境构建工作打下基础。

2 问题与模型

随着深度神经网络的加入，强化学习（RL）最近取得了重大进展，然而，很少有研究将 RL 应用于与客户互动的大型在线系统等物理世界任务，在这些场景中直接应用 RL 的一个主要障碍是，当前的 RL 算法通常需要与环境进行大量交互，这需要很高的物理成本，例如真实的金钱、从几天到几个月的时间、糟糕的用户体验。本篇论文主要是基于强化学习在电子商务搜索引擎排名中的应用 [4] 这份工作，对模拟环境构建问题进行的探讨。

2.1 问题与概述

淘宝的搜索引擎处理数十亿商品的毫秒级响应，而顾客对商品的偏好也丰富多样。从引擎的角度来看，淘宝平台的工作原理如下：一位客户前来并向搜索引擎发送搜索请求。然后，引擎通过对相关商品进行分类并向客户显示页面视图（PV）来对请求做出适当的响应。客户根据 PV 以及买家自己的意图给出反馈信号，例如购买东西、转到下一页或离开淘宝。搜索引擎接收反馈信号并对下一个 PV 请求做出新的决定。由于反馈信号，即客户行为，受到先前 PV 的影响，在优化搜索引擎策略时，将其视为多步决策问题而不是一步监督学习问题更为合理。淘宝的商业目标之一是通过优化展示 PV 的策略来实现销售额的最大化。因此，实际开发中将搜索引擎视作代理，顾客的反馈作为相应的环境，淘宝商品搜索是一个序列决策问题。可以合理地假设客户只能记住有限数量的 m 个最新 PV，这意味着反馈信号只受搜索代理的 m 个历史动作的影响。

另一方面，如果将客户视为代理人，将搜索引擎视为环境，那么客户的购物过程也可以被视为一个连续的决策过程。客户对排名商品的反应，即搜索引擎的动作。由于客户的行为，即反馈信号，受到最后 m 个 PV 的影响，这些 PV 由搜索引擎生成，并受到来自客户的最后反馈的影响。顾客的行为也具有马尔可夫性质。为顾客制定购物策略的过程可以看作是优化顾客在淘宝购物偏好的过程。

从上面的描述中可以看到，一方面，和传统的强化学习不同，虚拟淘宝中的客户和搜索引擎都具有马尔可夫性质，是彼此的代理，也是彼此的环境，二者的策略是耦合在一起的。另一方面，淘宝用户数量巨大，不同的用户拥有不同的喜好，这显然比一般的强化学习更为复杂。

2.2 系统模型

论文提出的虚拟淘宝模型架构大致如图 1所示：

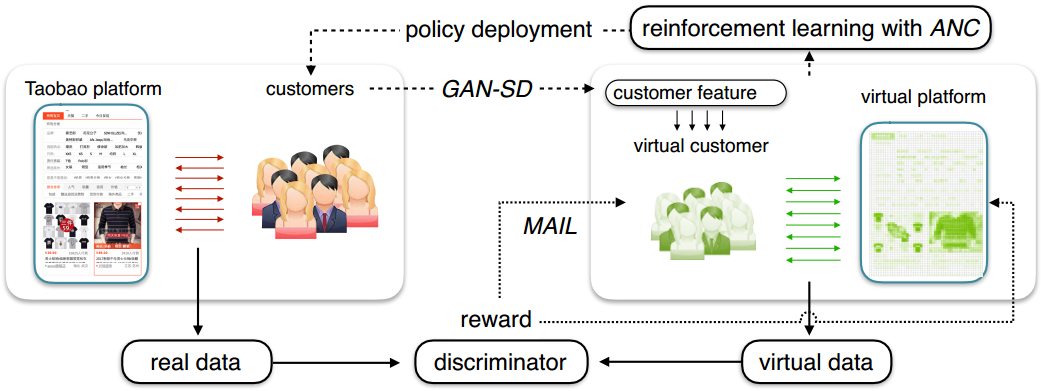

图 1. 虚拟淘宝模型图

首先，利用 GAN-SD 来学习生成客户特征。然后利用客户特征，通过 MAIL 方法学习客户的行为，生成交互。在训练了虚拟淘宝环境之后，平台策略在具有 ANC 的虚拟淘宝中进行训练。最后，平台引擎代理直接部署在真实的淘宝上。

3 复现工作

按照上述描述，我将复现过程大致分为三个阶段：首先复现 GAN-SD 算法，用于改进客户特征的生成，使其更好地匹配真实分布。接着通过复现 MAIL 方法验证对抗性学习对强化学习行为策略预测的有效性。最后，为了避免过度拟合虚拟淘宝环境的缺陷，我复现了 ANC 策略，用于规范策略模型，确保其在真实世界中具有良好的性能。由于没有源代码可供参考，因此在下面的复现工作中，我是在伪代码基础上进行的复现。

3.1 平台配置

下表 1是复现代码运行的环境配置和基本说明。

表 1. 环境配置

编程语言	Python
软件平台	Windows11 操作系统；PyCharm；
硬件环境	12 核 Intel i7 CPU；32GB 内存；240GB 固态硬盘
其他说明	Python>3.11;gym>=0.10.5;torch>=0.4.0 。虚拟淘宝是基于 PyTorch 进行训练和开发的。模型训练完成之后注册到 Gym 中，并利用 DDPG 提高的强化学习代理进行实验验证。

3.2 GAN-SD 代码复现

GAN-SD 主要用于生成客户特征。为了构建虚拟淘宝，需要首先生成客户特征，即对一个用户进行采样，该用户包括来自页面的请求，以触发交互过程。生成的客户分布应该与真实的客户分布类似。学习高维空间中的分布是具有挑战性的。高斯混合模型（GMM）等经典方法难以很好地近似分布。众所周知，生成对抗性框架旨在生成接近原始数据的样本，并在生成图像方面取得了巨大成功。然而，由于传统的 GANs 鉴别器决定实例是否来自真实，因此它们缺乏捕获客户的分发架构的能力。为了生成分布而不是单个实例，作者提出了用于模拟分布的生成对抗性网络（GAN-SD）：它的判别器采用的仍然是标准损失函数：二元交叉熵损失函数；而它的生成器损失函数引入了熵项、KL 散度项，具体公式如下：

$$E_{p_{\mathcal{D}},p_{\mathcal{G}}}[D(G(z)) + \alpha\mathcal{H}(V(G(z))) - \beta KL(V(x)||V(G(z)))]. \tag{1}$$

熵项 $\alpha\mathcal{H}(G(z))$：这个项鼓励生成器产生多样化的输出。熵项促使生成器的输出分布更加均匀，从而生成多样性更高的伪造数据。

KL 散度项 $\beta\mathrm{KL}(x\|G(z))$：这个项衡量了生成器产生的伪造数据与真实数据分布之间的差异。KL 散度用于惩罚生成器产生与真实数据分布差异较大的伪造数据。

利用 KL 散度和熵约束，GAN-SD 从真实数据中学习具有更多引导信息的生成器，并且可以生成比传统 GAN 更好的分布。我们利用论文提供的公式对生成器损失函数进行编写，具体代码及其注释如下图 2所示：

```
# 自定义生成器的损失函数
 Administrator
def generator_loss(fake_outputs, real_data, noise):
    # 计算熵项
    entropy_loss = -torch.mean(torch.log(fake_outputs) * fake_outputs)

    # 计算KL散度项
    kl_loss = torch.mean(torch.log(fake_outputs) * fake_outputs - torch.log(real_data) * real_data)

    # 生成器损失包括对抗损失、熵约束和KL散度项
    loss = torch.mean(-torch.log(fake_outputs)) + alpha * entropy_loss - beta * kl_loss
    return loss
```

图 2. 生成器损失函数代码

基于上面定义的损失函数，利用 GAN 网络能够生成非常真实的用户特征。为了方便后续的使用，我们训练之后生成 pt 模型文件保存，如下图 3所示：

```python
# 用户模型
3 usages
class UserModel(nn.Module):
    """
    用户模型的构建：用户在虚拟淘宝中就是环境模型。
    该模型继承了nn.Module：PyTorch中构建的神经网络模型的基类
    """

    def __init__(self, instance_dimesion=88, seed_dimesion=128, n_hidden=128, learning_rate=0.001):...

    def generator(self, z):...

    1 usage
    def softmax_feature(self, x):...

    def generate(self, z=None):...

    def load(self, path=None):...
```

图 3. 用户模型代码

编写 UserModel.py 来加载该模型进行用户特征的生成。其中 load() 方法用于加载 pt 保存的模型，而调用 generator() 方法则可以生成仿真的用户特征。上述代码有详细的注释，考虑到篇幅限制因此对于非算法代码并不在此进行详细展开。

3.3 MAIL 代码复现

通过模拟客户策略，生成客户与平台之间的互动，这是虚拟淘宝的关键。我们知道，GAIL（生成对抗性模仿学习）[3] 允许代理与环境交互，并通过 RL 方法学习策略，同时在训练过程中改进奖励函数。遵循 GAIL 的思想，作者提出了多智能体对抗性模仿学习（MAIL）方法来实现这一点。与在静态环境中训练一个代理策略的 GAIL 不同，MAIL 是一种训练客户策略和引擎策略的多代理方法。通过这种方式，学习到的客户策略能够概括为不同的引擎策略。由于 MAIL 将这两种策略一起训练，即代理和环境，因此它只需要历史数据，不需要访问真实的环境。奖励函数被设计为生成的数据和历史状态-动作对的非歧视性。为了运行 MAIL，需要历史轨迹 τ_e 和客户分布 P^c，这是 T_σ^c 所要求的。在前面的复现过程中，$\mathcal{P}^c$ 是通过 GAN-SD 预先学习的。初始化变量后，我们开始 MAIL 的主要过程：在每次迭代中，收集客户代理和环境之间交互过程中的轨迹（第 4-9 行）。然后，从生成的轨迹中进行采样，并通过梯度方法优化奖励函数（第 10-11 行）。然后，κ, τ 通过使用一个 RL 方法优化 $\mathcal{M}^c$ 中的联合策略 $\pi_{\kappa,\sigma}^c$ 进行更新；（第 12 行）。当迭代终止时，MAIL 返回客户代理策略 π^c.。它的具体运行伪代码如下图 4所示：

Algorithm 2 MAIL

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

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

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

图 4. MAIL 伪代码

按照上述伪代码进行了复现工作，首先是通过 GAN-SD 预先学习得到客户分布然后执行 MAIL 算法，如下图 5所示：

```
for i in range(I):
    for j in range(J):
        tau_j = []  # 存储轨迹
        s_c = sample_state_from_customer_distribution()
        a_c = policy_theta(s_c, kappa)

        while not TERMINATED:
            a_c = policy_theta(s_c, kappa)
            tau_j.append((s_c, a_c))
            s_c = environment_dynamics(s_c, a_c)

        # 步骤 10-11：优化奖励函数
        theta = optimize_reward_function(theta, tau_j, P_c)

    # 步骤 12：通过RL方法优化客户策略和引擎策略
    kappa, sigma = optimize_joint_policy(kappa, sigma, P_c)

# 输出最终的客户策略
final_customer_policy = policy_theta(s_c, kappa)
```

图 5. MAIL 复现代码

MAIL 其实就是基于经典的 GAIL 进行的改进，通过环境和智能体的对偶性和对抗学习同时训练环境模型和策略模型。它在每次迭代中收集客户代理和环境之间交互过程中的轨迹，接着，从生成的轨迹中进行采样，并通过梯度方法优化奖励函数。迭代过程中不断优化客户策略和引擎策略。

3.4 ANC 复现

如果部署的策略与历史策略相似，虚拟淘宝的行为往往比两种策略完全不同的情况更相似。然而，类似的策略意味着没有什么改善。强大的 RL 算法，如 TRPO，很容易地训练代理过拟合虚拟淘宝，这意味着它在虚拟环境中表现良好，但在现实环境中表现不佳。事实上，我们需要在准确性（即类似于历史政策）和改进（即远离历史政策）之间进行权衡。然而，由于历史引擎政策在实践中不可用，无法衡量当前政策与历史政策之间的距离。作者提出了行动规范约束（ANC）策略来控制所采取行动的规范。也就是说，当所采取行动的规范大于大多数历史行动的规范时，会对代理进行处罚：$r'(s,a) = \dfrac{r(s,a)}{1+\rho\max\{\|a\|-\mu,\,0\}}$ 。它的大致代码逻辑如下图 6所示：

```python
# 原始的奖励函数
1 usage
def original_reward_function(s, a):
    # 根据任务定义的原始奖励函数
    return np.log(policy_theta(s, a, theta)) - np.log(1 - policy_theta(s, a, theta))

# ANC 修改后的奖励函
# 定义 ANC 参数
rho = 0.1  # 超参数
mu = 0.5   # 超参数
# ANC 修改后的奖励函数

def modified_reward_function_with_anc(theta, tau_j, P_c):
    for t in range(len(tau_j)):
        s_c, a_c = tau_j[t]

        # 获取历史动作的范数的最大值
        max_norm = compute_max_norm_of_historical_actions()

        # 计算 ANC 调整项
        anc_adjustment = 1 / (1 + rho * max(0, np.linalg.norm(a_c) - mu))

        # 计算修改后的奖励
        r_modified = original_reward_function(theta, s_c, a_c) * anc_adjustment
```

图 6. ANC 复现代码

也就是说，首先获取历史动作的范数的最大值，然后计算 ANC 调整项，最后得到修改后的奖励。

3.5 项目搭建

在对上面的算法进行具体的复现之后，之后我们进行训练生成模型，然后将模型保存起来构成一个虚拟淘宝供外部调用，项目代码的整体结构如下图 7所示：

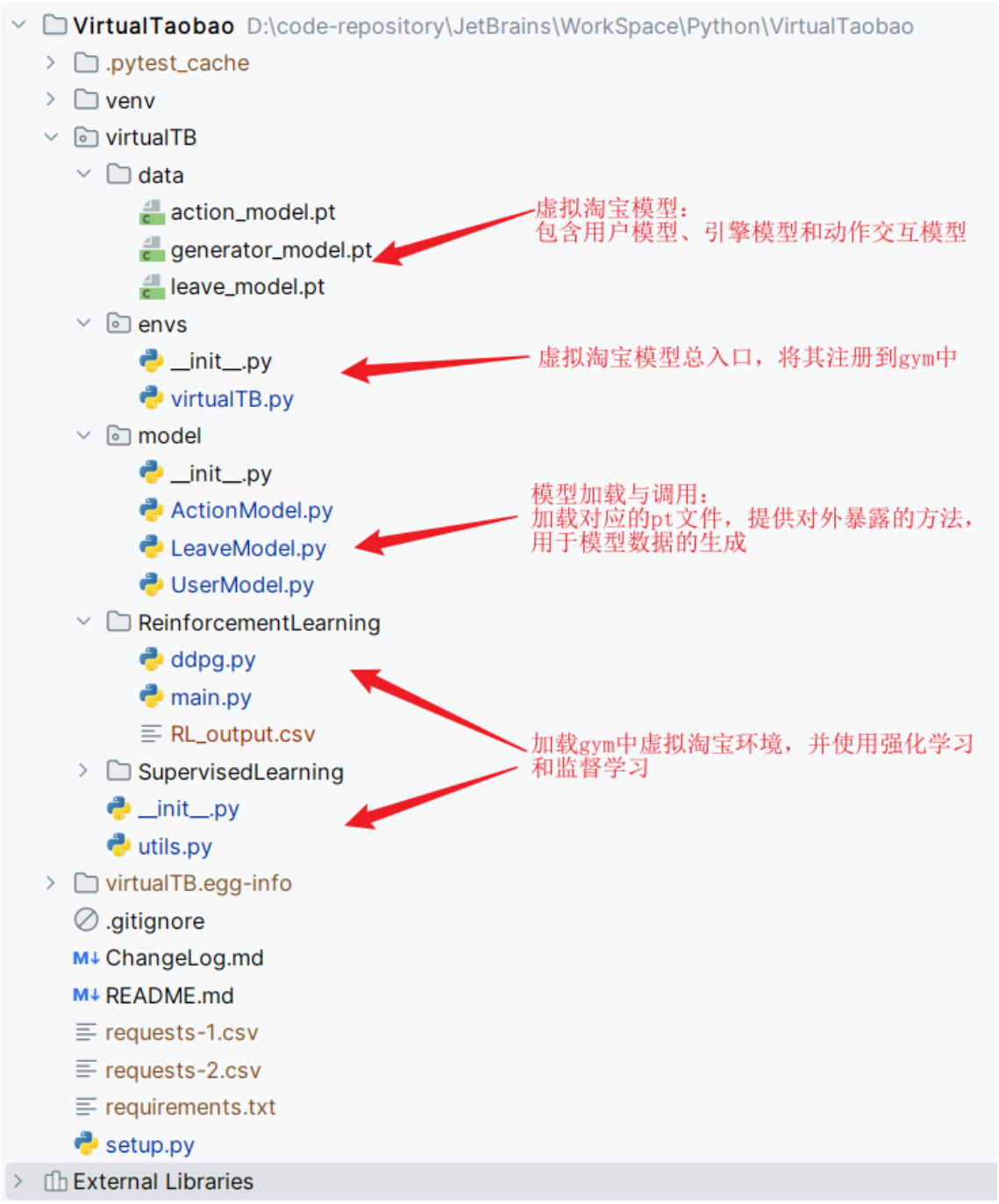

图 7. 项目代码整体结构说明

根据上面复现的算法，我们构建了用于生成用户特征的用户模型 generator.pt 文件、根据用户特征生成返回页的引擎模型 leave_model.pt 文件、用于动作生成的 action_model.pt 文件。然后我对这三个 pt 文件编写了对应的类保存在 model 包中。它们加载 pt 文件并向外提供模型生成的数据。data 包和 model 包共同构成了虚拟淘宝的基本架构。之后在 envs 包中将这个自定义的环境注册到 gym 中。在后续的实验环境搭建中，利用 gym 将其加载出来，作为环境，提供给 DDPG 代理和 RL 来进行对比性实验。

4 实验与分析

4.1 实验环境搭建

如何评价算法性能? 我认为 CTR(点击通过率) 是一个非常直观的指标。也就是说，一个优秀的搜索引擎应该保证用户尽快找到理想的商品，而不是一直和引擎进行点击交互。

出于用户信息隐私保护和安全性，淘宝用户数据集的收集十分困难，因此，我按照作者提供的比较有效的匿名淘宝用户数据集（包含客户特征、项目特征、标签和点击次数）实验来验证在虚拟淘宝中训练出来的代理策略可以有明显优于传统的监督方法的在线性能。首先引入了 DDPG 提供的深度强化学习代理和经典的监督学习，并使用构建好的虚拟环境来进行训练，收集每一次训练之后的 CTR(点击通过率) 来比较训练效果。然后编写代码将生成的数据集进行可视化。引入 DDPG 强化学习和经典的监督学习，具体代码如下图 8所示：

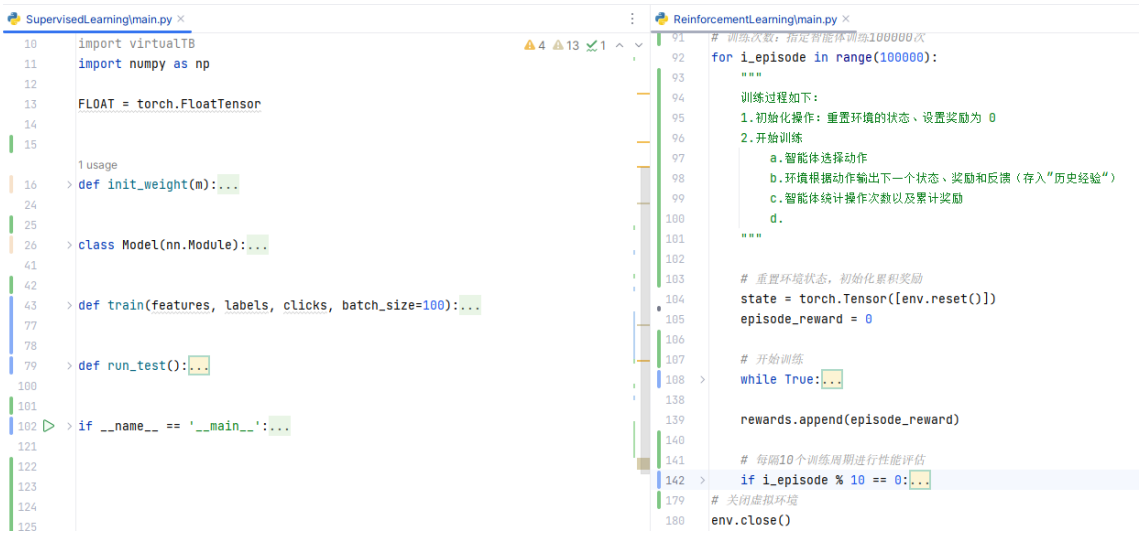

图 8. DDPG 和传统监督学习

左图引入 DDPG 强化学习算法，加载 gym 中预定义的虚拟淘宝模型进行代理策略实验。右图使用经典的监督学习进行推荐。

4.2 结果分析

本实验主要是验证利用在虚拟淘宝中训练出来的策略可以有明显优于传统的监督方法的在线性能。结果如图 9所示：

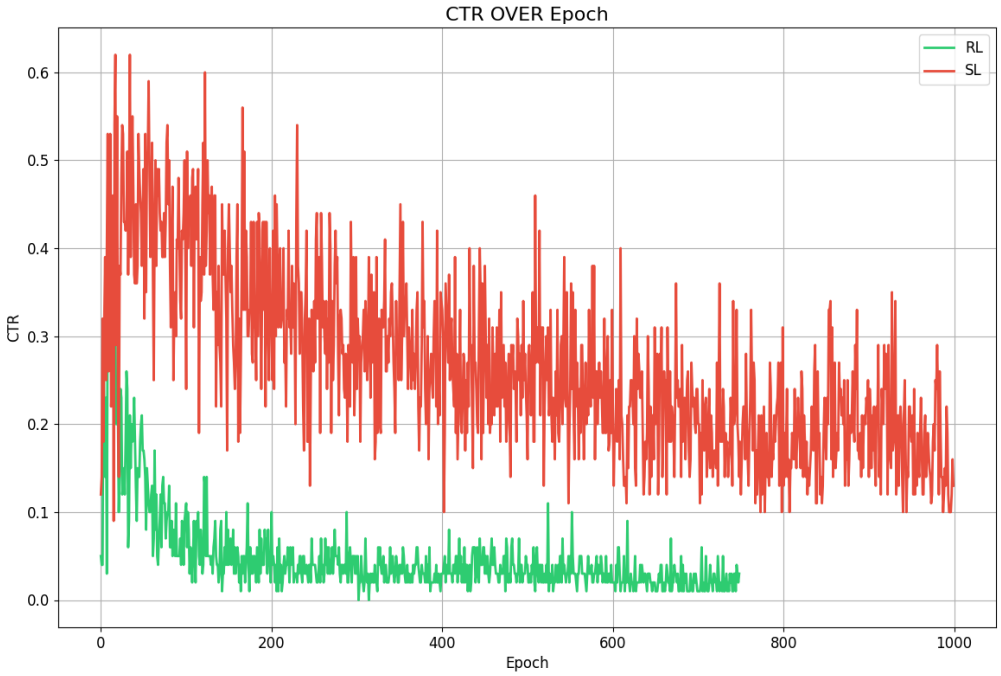

图 9. 实验结果示意

我们可以看到，DDPG 在该虚拟环境中的 CTR 点击通过率在很短的时间内就实现收敛。也就是说，在该环境中训练的代理能够很快的推荐出让用户满意的结果，使得点击率快速下降并收敛。借助该虚拟环境的训练，相比于监督学习，强化学习能更短时间内实现更好的效果，在线性能更优异。

5 后续工作与拓展

虚拟淘宝相比于传统的强化学习不同之处在于：1. 搜索引擎和用户各自为彼此的环境，它们的策略耦合。2. 用户数量巨大，不同的用户具有不同的特征。尽管它比以往的强化学习物理环境构建要复杂许多，但是仍给予了我当前的工作许多启发。在小组汇报之后，我尝试将这思路应用到云计算资源缩放的强化学习应用中，并根据实际情况进行了一部分的开发和改进。

在我目前的科研工作中，我已经编写好了用于服务请求的客户端脚本、提供服务的 web 服务以及用于服务指标监听和获取的监听器。首先是将 web 服务打包成容器镜像并部署到集群之中，然后利用服务请求脚本读取请求数据集向集群施加负载，集群采用原生的 HPA 机制进行资源缩放，我使用开发的监听器读取指标来收集强化学习环境变化并生成样本数据集。之后尝试利用生成的样本数据集和虚拟淘宝算法构建虚拟云环境，用于强化学习代理的预训练。

在后续复现工作过程中我也遇到了如下问题，MAIL 是通过环境和智能体的对偶性和对抗学习同时训练环境模型和策略模型。在现实世界的应用中，离线数据的收集往往带有选择偏差，即对于每种状态，每个动作都可能被不公平地选择。上面的环境模型学习的方法并不直接处理模型学习的优化目标对环境模型的影响而是通过限制高风险区域的策略探索和学习

来解决这个问题（经验风险最小化）。当收集具有选择偏差的数据集时，状态和行为之间的高度相关性使下一个状态的因果关系不可知，这最终导致模型预测对反事实数据的泛化能力较差。因此在离线环境中获得对抗性反事实数据分布是不可行的。为了解决上述问题，在老师的建议下，尝试使用 Xiong-Hui Chen [2] 的工作来进行改进，实现对抗性加权经验风险最小化 (AWRM) 的目标。对于每次迭代，AWRM 采用对抗性策略构建对抗性反事实数据集，最大化模型的预测误差，并驱动模型降低对抗性反事实数据分布下的预测风险。

从近似数据分布中学习环境模型,算法名为 Generative Adversarial offLIne counterfactuaL Environment model learning (GALILEO) 通过生成对抗式的方案完成模型的学习。它的原理是通过判别器来估计一个逆倾向得分 (inverse propensity score ; IPS) 来重新加权训练数据（事实与反事实），以近似随机策略下的数据分布，并在此重新加权的分布下训练模型。

GALILEO 可以视为是对 MAIL 构建模拟环境的一个补充改进，它有效的解决了模拟环境学习过程中存在的选择性偏差，驱动模型降低对抗性反事实数据分布下的预测风险。

6 总结与展望

在选择虚拟淘宝这篇文章之前，我调研了大量深度强化学习及其环境模拟的论文，了解了当前构建物理环境模拟的主要解决方案和它们的缺陷。虚拟淘宝基于生成对抗模仿学习 (GAIL) 进行了改进，提出了多智能体对抗性模仿学习 (MAIL)。它克服了传统方法行为克隆的脆弱性以及逆强化学习的昂贵性，为虚拟环境的构建提供了新的思路。实验证明，使用该方法构建的模拟环境中进行代理策略训练是非常有效。

虚拟淘宝的算法结构并不复杂。我按照它的系统模型将复现工作具体分为了 3 个工作：GAN-SD 算法复现；MAIL 算法复现；添加 ANC 约束规范。由于论文给出了每个算法的伪代码，因此我的复现工作主要是对三个核心伪代码的复现。之后我将训练好的模型打包成 pt 文件并保存起来构建项目供他人使用。最后我采用监督学习和强化学习对比实验来验证模拟环境的性能。值得一提的是，在正式实验开展前我还对上面三个核心复现工作进行了比较来验证性能。例如，GAN-SD 和标准 GAN，MAIL 和 GAIL，有无 ANC。经过实验比对发现，虚拟淘宝提出的算法 GAN-SD 和 MAIL 显著优化了虚拟淘宝的性能。不过比较遗憾的是，由于部分数据集的缺失导致 ANC 无法进行验证，因此未作详细的探究。之后我根据提供的有限数据集采用 CTR 作为指标进行了 DDPG 和监督学习在模拟环境中的性能对比实验。实验结果显示 DDPG 代理在模拟环境中能实现快速收敛，说明了论文提出的算法模型的有效性。

复现过程并不是一帆风顺的，我对很多观点和方法论仍然停留在一知半解中。例如，对于 GAN-SD 中的损失函数，我始终不太明白为什么要引入熵项和 KL 散项的 GAN-SD 算法会在此场景下优于标准损失函数。在后续云计算模拟环境搭建中我也进行了 GAN-SD 与 GAN 标准顺势函数对比实验，实验结果证明 GAN-SD 仍然优于 GAN 标准损失函数。我希望在后续的工作中能找到原因。除此之外，在后续工作和拓展实验中，我还发现虚拟淘宝不能有效地应对反事实数据，进而造成选择偏差。为此，我和导师、同门进行了密切的交流和细致的探讨，最后找到合理的解决方案。最后是数据集生成的问题，由于场景的迁移，因此我需要重新生成新的数据集用于云资源虚拟环境的训练，这意味着我需要找到企业级的服务请求数据集和开发脚本和服务，并将它们部署到集群中，监控服务的缩放指标，生成符合条件的数据集。整个过程虽然耗费了我大量的时间，但确实增强了我对云计算的认识，也加深了我对

虚拟淘宝中数据集处理代码的理解。

在前沿技术课程的复现工作中，我收获了很多。从发现问题到查找解决方案再到发现新的问题再到交流思路，这不断增强了我的编程能力还提高了我发现问题、解决问题的能力。虚拟淘宝这篇文章选自在 AAAI 会议的，在阅读这篇论文前我还对相关工作进行了详细的调研，这个调研工作使我了解了当前物理环境采样成本过高问题的几个解决方案思路和它们的缺陷（行为克隆的脆弱性以及逆强化学习的昂贵性 [1]），并认识了 GAIL 对抗式模仿学习。虚拟淘宝遵循 GAIL 的思想，针对淘宝自身的特性，即用户策略会收到引擎的响应影响，作者提出了多智能体对抗性模仿学习（MAIL）方法来实现这一点。另外虚拟淘宝还根据实际应用场景对 GAN 也进行了改进。通过阅读、复现这篇论文，让我知道了一个优秀的论文应该如何编写，它解决问题的创新性思维方式和总结归纳问题、方法的科学严谨精神让我深受启发。

总体来说，借助这篇文章的复现工作，我了解了虚拟环境构建的要素、步骤以及它的方法论，为我后续科研打下了比较坚实的基础。

参考文献

[1] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*, 2016.

[2] Xiong-Hui Chen, Yang Yu, Zheng-Mao Zhu, Zhihua Yu, Zhenjun Chen, Chenghe Wang, Yinan Wu, Hongqiu Wu, Rong-Jun Qin, Ruijin Ding, and Fangsheng Huang. Adversarial counterfactual environment model learning. *arXiv preprint arXiv:2206.04890*, 6 2022.

[3] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[4] Yujing Hu, Qing Da, An Zeng, Yizhou Yu, and Yi Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.