Deep Reinforcement Learning for List-wise Recommendations

基于强化学习的推荐系统（京东）

Conference'17, July 2017, Washington, DC, USA

报告人：

2018-3-15

摘要

- 1.以往的推荐系统往往是基于固定的策略和通过静态的算法给出用户建议；
- 2.本文提出一种新型的推荐系统，能够通过与用户进行不断交互的方式持续的改进自身的策略。
- 3.根据马尔科夫决策过程将用户与智能体之间建立联系，采用强化学习算法根据用户对于推荐物品的反馈不断地调整算法性能，最终得到最优推荐策略；
- 4.文中给出了一种在线的“用户-智能体”交互模拟器，能够在离线的状态下对模型的参数进行预训练和评估。
- 5.设计LIRD推荐框架用于对推荐结果进行排序，并使用真实的电子商务数据对模型效果进行验证。

List-Wise Recommender System

Actor-Critic

Deep Reinforcement Learning

Online Environment Simulator

目录 CONTENTS

论文导读

Introduction

PART ONE

- 1.1List-wise Recommendation
- 1.2Architecture Selection
- 1.3Online Environment Simulator
- 1.4Our Contributions

绪论

Introduction

传统的推荐系统往往是采用固定的策略或者遵循最大贪婪方式为用户推荐产品，这种方法往往缺乏灵活性，而且在推荐上面往往只考虑当前的回报，忽视了长期的回报。

Introduction

基于深度学习的推荐系统

模型优点

1.可以自动化学习用户行为，合理的捕捉用户与推荐系统之间的交互，从而进而产生合适的推荐策略

模型优点

2.其次，该推荐系统注重一个长期的累计回报，根据累计收益合理的进行推荐

Reinforcement Learning

强化学习

01

局限性

传统的MDP以及Q-Learning往往通过存储value的形式进行推荐，因此在数量级较大的推荐系统中无法灵活的进行使用

02

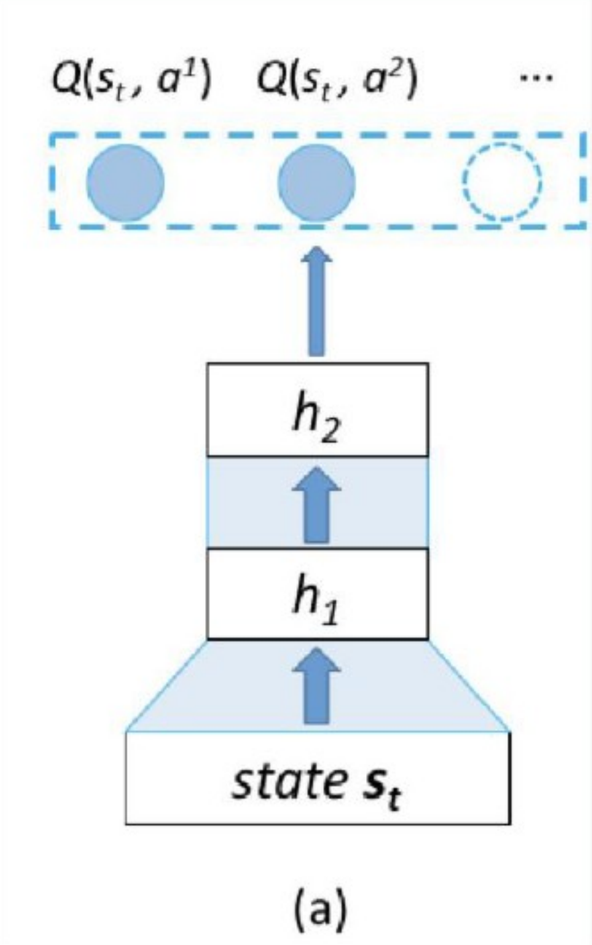
解决方案

因此采用人工神经网络非线性的拟合action的计算过程，对于大量数据集的推荐具有更好的支持

1.1List-wise Recommendation

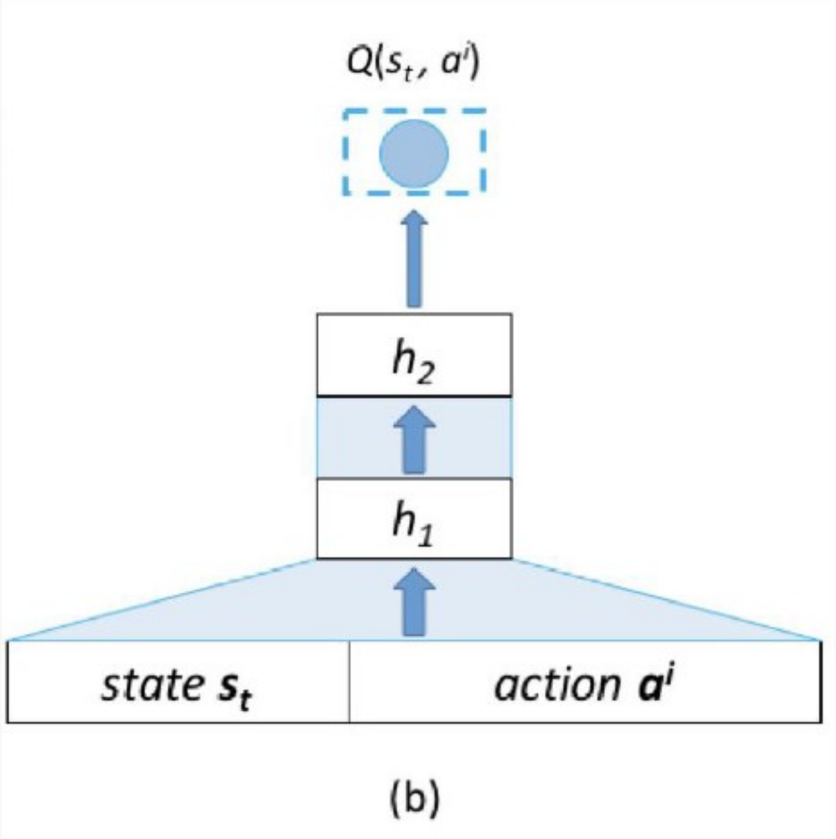
- 推荐系统往往通过列表的形式给用户列出一系列的推荐建议，能够更加灵活多样的给以用户选择。
- 对于列表式的推荐，论文结合强化学习给以用户一种列表式的action选择，每一组action都是多个子物品共同的集合。
- DQN可以单独计算所有离散的推荐物品的Q值，并选择Q值较高的物品构成推荐列表，但是这样往往会导致推荐同一个相似或者互补的产品，而推荐的过程中往往推荐一些互补的产品可能会获得更好的效果，因此文中提出的方法，通过采用一定的规则或者推荐关系生成一系列的互补产品来加强推荐系统的性能。

1.2Architecture Selection



- 输入state输出所有action的Q-Value,这种模型适合高state空间和小的action空间，如Atari; 不能够处理大的以及动态变化的action，比如电子商务的推荐系统；

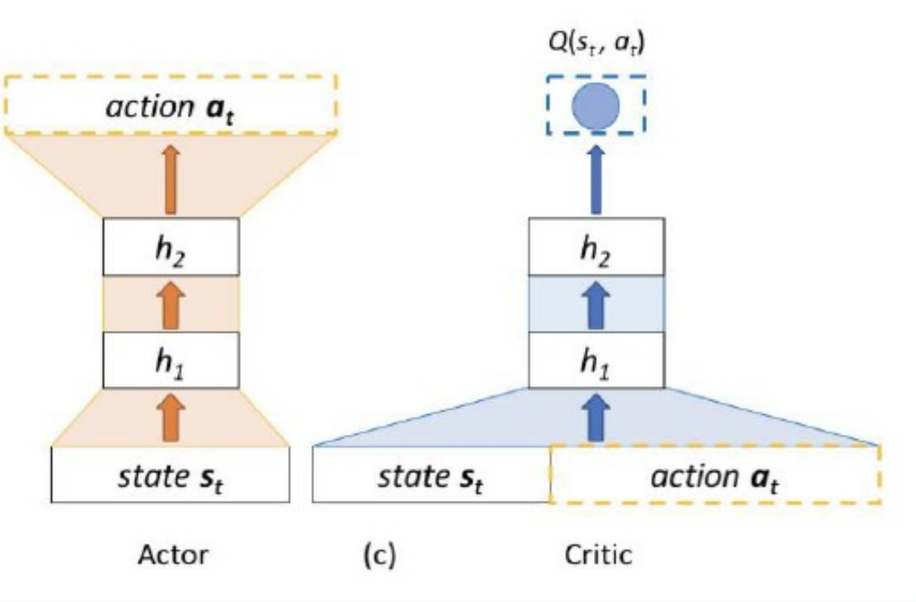
1.2Architecture Selection



(b)

- 针对state和action作为神经网络的输入，直接输出Q-Value这种网络结构不需要在内存中存储每一个action对应的Q-Value，因此可以处理非常庞大的action空间、甚至是连续的动作，但是这种结构的时间复杂度较高，因为需要单独计算所有潜在的Q(state, action)值。

1.2Architecture Selection



- 为了解决上述两种问题提出了建立在Action-Critic上的推荐框架
- Actor用于输入当前的状态并旨在输出当前状态下较高action
- Critic使用价值函数根据state和Actor给出的action计算当前的Q值，这是对当前state所选action是否匹配最优action的一个判断，Critic网络采用跟b图相同的网络结构
- Actor根据Critic的判断，更好的提高自己的性能，输出最优策略
- 这种架构适合大型的action空间，而且减少了计算的冗余

1.3Online Environment Simulator

- 在线游戏Atari可以根据当前采取的行动，及时的获得反馈，但是在线推荐很难获得立即的奖励。因此有必要建立离线的预训练机制，并将其用于评估模型中去。
- 论文提出了一个在线的环境模拟器（Online Environment Simulator）输入当前的动作，并输出模拟的在线奖励，使得AC框架能够对参数进行拟合，从而改进推荐系统。
- 文中提出的这种基于用户历史数据的模拟器，根据历史记录推荐给用户的产品，用户对这些推荐项目做出的动作，通过用户点击和购买记录得到及时的奖励，根据回报对模型进行训练和评估，因此可以通过上述过程进行探索和利用，最终得到效果较好的强化学习推荐系统。

1.4Our Contributions

- 1.提出了在线模拟器用于离线训练参数模型；
- 2.**LI**st-wise **R**ecommendation framework based on **D**eep reinforcement learning （ LIRD ）可以用于大型的项目推荐并且减少了多余的计算量；
- 3.使用真实的电商数据验证了模型的有效性和准确性；

模型框架
Framework

PART TWO

- 2.1Problem Statement
- 2.2Online User-Agent Interaction Environment Simulator
- 2.3The Actor Framework
- 2.4The Critic Framework
- 2.5The Training Procedure
- 2.6The Testing Procedure

2.1Problem Statement

01

目标对象：

Recommender Agent (RA)

02

环境：

User/模拟器

03

性质：

符合马尔科夫决策(MDP)过程

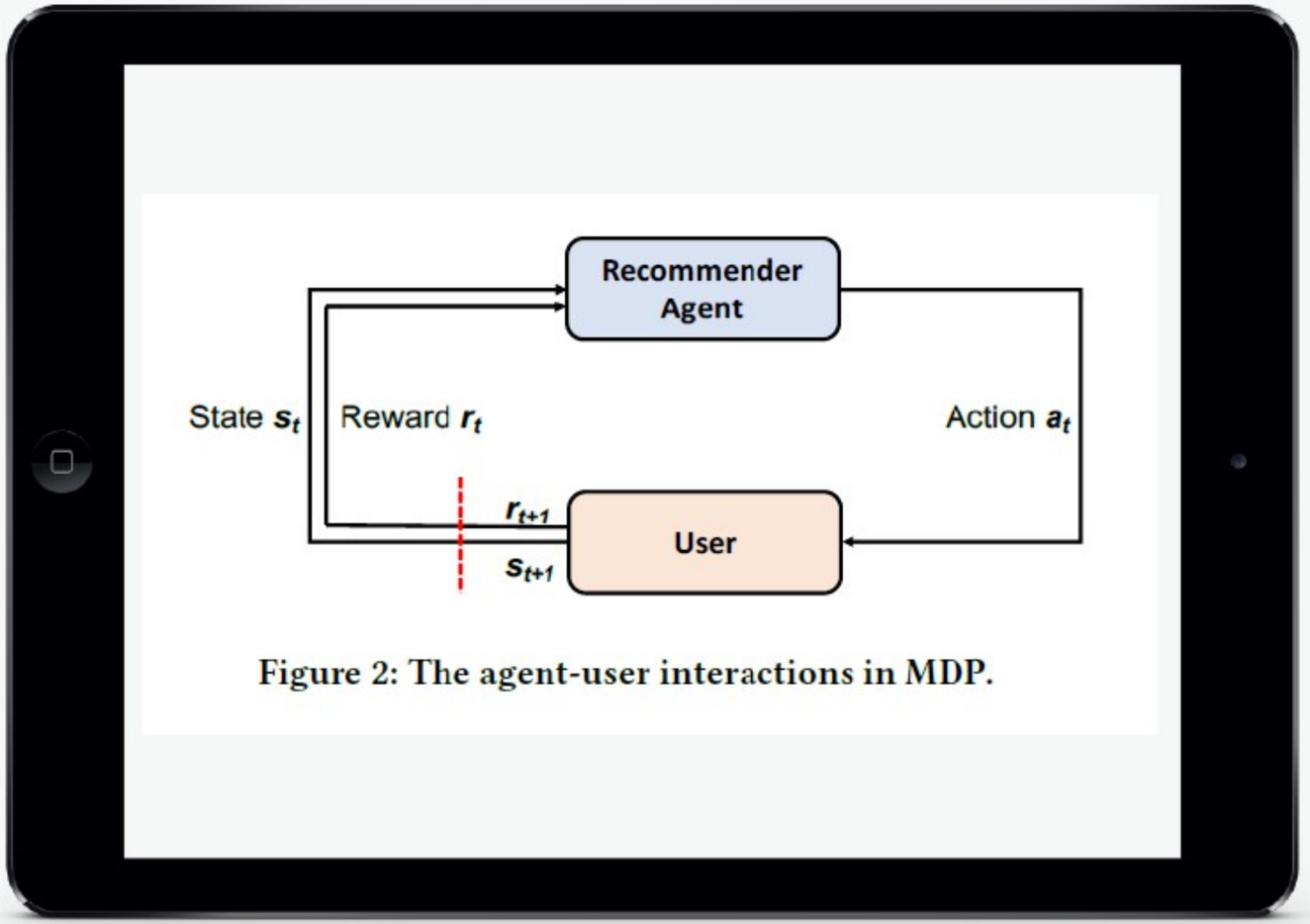
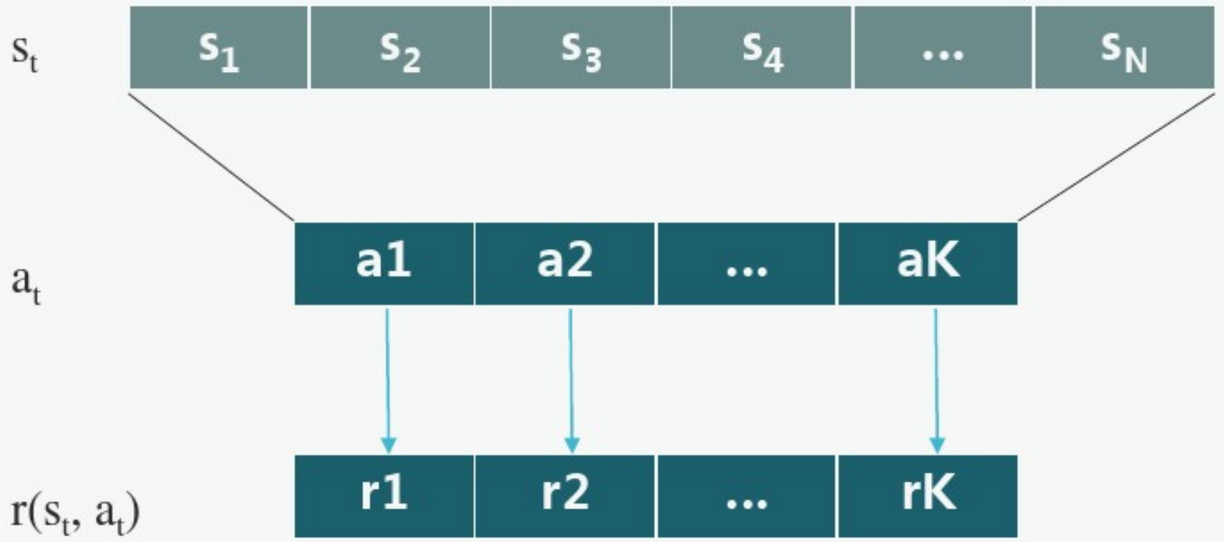


Figure 2: The agent-user interactions in MDP.

2.1Problem Statement

$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

- **State space $\mathcal{S}$**: A state $s_t = \{s_t^1, \cdots, s_t^N\} \in \mathcal{S}$ is defined as the browsing history of a user, i.e., previous N items that a user browsed before time t. The items in s_t are sorted in chronological order.
- **Action space $\mathcal{A}$**: An action $a_t = \{a_t^1, \cdots, a_t^K\} \in \mathcal{A}$ is to recommend a list of items to a user at time t based on current state s_t, where K is the number of items the RA recommends to user each time.
- **Reward $\mathcal{R}$**: After the recommender agent takes an action a_t at the state s_t, i.e., recommending a list of items to a user, the user browses these items and provides her feedback. She can skip (not click), click, or order these items, and the agent receives immediate reward $r(s_t, a_t)$ according to the user's feedback.



2.1Problem Statement

$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

- **Transition probability** $\mathcal{P}$: Transition probability $p(s_{t+1}|s_t, a_t)$ defines the probability of state transition from s_t to s_{t+1} when RA takes action a_t. We assume that the MDP satisfies $p(s_{t+1}|s_t, a_t, ..., s_1, a_1) = p(s_{t+1}|s_t, a_t)$. If user skips all the recommended items, then the next state $s_{t+1} = s_t$; while if the user clicks/orders part of items, then the next state s_{t+1} updates. More details will be shown in following subsections.
- **Discount factor** γ: $\gamma \in [0, 1]$ defines the discount factor when we measure the present value of future reward. In particular, when $\gamma = 0$, RA only considers the immediate reward. In other words, when $\gamma = 1$, all future rewards can be counted fully into that of the current action.

2.1Problem Statement

- 此外我们也要考虑品牌、时间、价格以及日期等对推荐的产品进行排序
- 用户浏览的物品这里通过词嵌入的方式将其装换成特征向量，用一个低维稠密的向量进行表示，因此推荐系统的一个session可以看成一是个句子。

2.2Online User-Agent Interaction Environment Simulator

- 为了离线训练模型并对其进行评估，设计了在线的用户-智能体环境交互模拟器。
- 模拟器在推荐过程中给出当前的状态state，RA推荐一系列的item给用户，用户浏览这些item并给出相应的反馈
- 因此模拟器基于之前的用户行为，给出 ：$f\ (st\ ,\ at\ ) \rightarrow rt$
- 基于具有相同兴趣的用户总是在类似的物品上做出相似的决定（协同过滤技术），我们根据历史存在的state-action与当前的state-action匹配，从而模拟生成一个reward

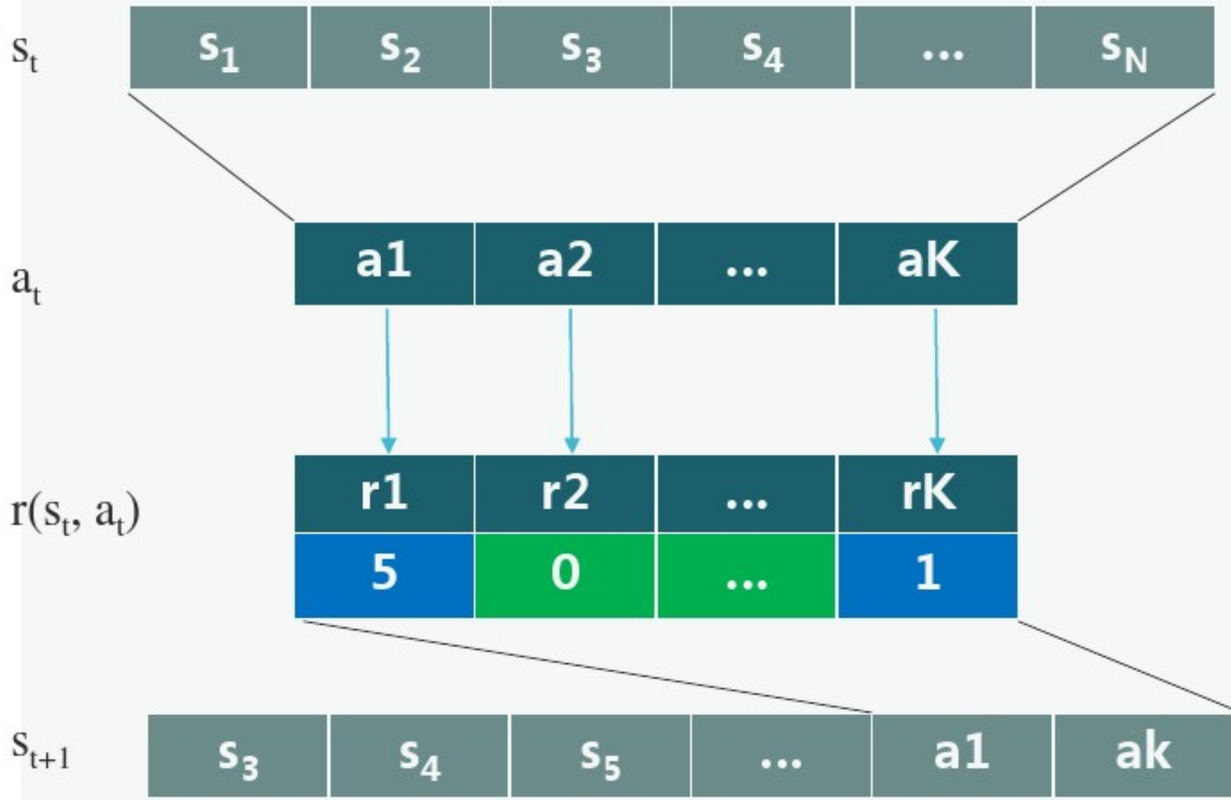
2.2Online User-Agent Interaction Environment Simulator

Algorithm 1 Building Online Simulator Memory.

Input: Users' historical sessions B, and the length of recommendation list K.

Output:Simulator Memory $\mathcal{M}$

for *session* $= 1, B$ **do**
  Observe initial state $s_0 = \{s_0^1, \cdots, s_0^N\}$
  for *item order* $l = 1, L; K$ **do**
    Observe current state $s = \{s^1, \cdots, s^N\}$
    Observe current action list $a = \{a_l, \cdots, a_{l+K-1}\}$
    Observe current reward list $r = \{r_l, \cdots, r_{l+K-1}\}$
    Add the triple $((s, a) \rightarrow r)$ in $\mathcal{M}$
    for $k = 0, K-1$ **do**
      if $r_{l+k} > 0$ **then**
        Remove the first item in s
        Add the item a_{l+k} in the bottom of s
      end if
    end for
  end for
end for
return $\mathcal{M}$



$$\mathcal{M} = \{m_1, m_2, \cdots\} \rightarrow ((s_i, a_i) \rightarrow r_i)$$

2.2Online User-Agent Interaction Environment Simulator

$$Cosine(p_t, m_i) = \alpha \frac{s_t s_i^\top}{\|s_t\|\|s_i\|} + (1-\alpha)\frac{a_t a_i^\top}{\|a_t\|\|a_i\|},$$

$$P(p_t \rightarrow r_i) = \frac{Cosine(p_t, m_i)}{\sum_{m_j \in \mathcal{M}} Cosine(p_t, m_j)},$$

- 例如给用户推荐两个物品，每个物品的回报如下：

滑动	点击	购买
0	1	5

- 因此这两个物品的最终回报会出现以下几种排列：

{(0, 0), (0, 1), (0, 5), (1, 0), (1, 1), (1, 5), (5, 0), (5, 1), (5, 5)},

- 计算两个行为的相似性：

2.2Online User-Agent Interaction Environment Simulator

◆ 计算回报：

$$P(p_t \rightarrow \mathcal{U}_x) = \frac{\sum_{r_i=\mathcal{U}_x} Cosine(p_t, m_i)}{\sum_{m_j \in \mathcal{M}} Cosine(p_t, m_j)}$$

$$= \frac{\alpha \frac{s_t}{\|s_t\|} \cdot \sum_{r_i=\mathcal{U}_x} \frac{s_i^\top}{\|s_i\|} + (1-\alpha) \frac{a_t}{\|a_t\|} \cdot \sum_{r_i=\mathcal{U}_x} \frac{a_i^\top}{\|a_i\|}}{\sum_{\mathcal{U}_y \in \mathcal{U}} \left(\alpha \frac{s_t}{\|s_t\|} \cdot \sum_{r_j=\mathcal{U}_y} \frac{s_j^\top}{\|s_j\|} + (1-\alpha) \frac{a_t}{\|a_t\|} \cdot \sum_{r_j=\mathcal{U}_y} \frac{a_j^\top}{\|a_j\|} \right)}$$

$$= \frac{\mathcal{N}_x \cdot \left(\alpha \frac{s_t \bar{s_x}^\top}{\|s_t\|} + (1-\alpha) \frac{a_t \bar{a_x}^\top}{\|a_t\|} \right)}{\sum_{\mathcal{U}_y \in \mathcal{U}} \mathcal{N}_y \cdot \left(\alpha \frac{s_t \bar{s_y}^\top}{\|s_t\|} + (1-\alpha) \frac{a_t \bar{a_y}^\top}{\|a_t\|} \right)}$$

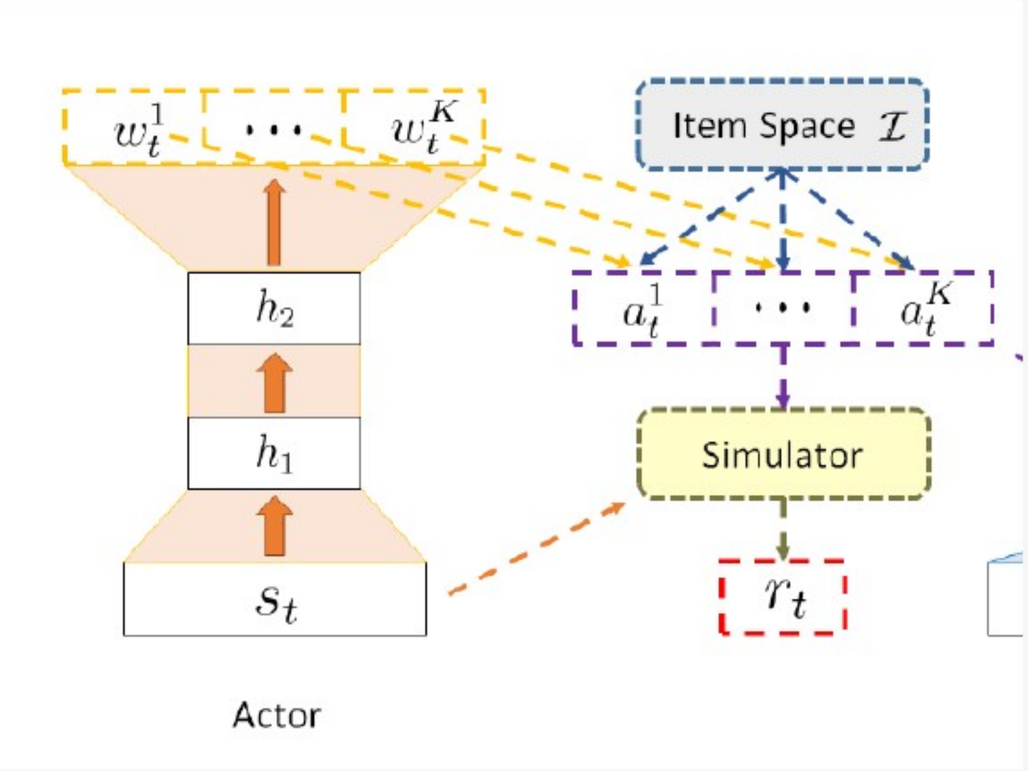
$$\bar{s_x} = \frac{1}{\mathcal{N}_x} \sum_{r_i=\mathcal{U}_x} s_i / \|s_i\| \qquad \bar{a_x} = \frac{1}{\mathcal{N}_x} \sum_{r_i=\mathcal{U}_x} a_i / \|a_i\|$$

2.2Online User-Agent Interaction Environment Simulator

- 计算回报： 实际中，回报往往不是用向量表示，而是用数值，通过匹配到的回报向量*Ux*，可以计算出回报数值。

$$r_t = \sum_{k=1}^{K} \Gamma^{k-1} u_x^k,$$

2.3The Actor Framework



Actor

已知当前的状态： $s_t = \{s_t^1, \cdots, s_t^N\}$

Actor的目标是求出向量wt: $f_{\theta^\pi} : s_t \rightarrow \mathbf{w}_t$

其中wt包含K个向量： $\mathbf{w}_t = \{\mathbf{w}_t^1, \cdots, \mathbf{w}_t^K\}$

通过与I中的物品相乘，得到一个分数score： $socre_i = \mathbf{w}_t^k \mathbf{e}_i^\top$.

通过选择前k个分数较高的物品，作为推荐的物品

2.3The Actor Framework

Algorithm 2 List-Wise Item Recommendation Algorithm.

Input: Current state s_t, Item space $\mathcal{I}$, the length of recommendation list K.

Output:Recommendation list a_t.

1: Generate $\mathbf{w}_t = \{\mathbf{w}_t^1, \cdots, \mathbf{w}_t^K\}$ according Eq.(5) $f_{\theta^\pi} : s_t \rightarrow \mathbf{w}_t$
2: **for** $k = 1, K$ **do**
3: Score items in $\mathcal{I}$ according Eq.(6) $socre_i = \mathbf{w}_t^k \mathbf{e}_i^\top$.
4: Select the an item with highest score as a_t^k
5: Add item a_t^k in the bottom of a_t
6: Remove item a_t^k from $\mathcal{I}$
7: **end for**
8: **return** a_t

2.4The Critic Framework

Critic的目的是根据当前S下做出的Action给出一个分数Q（s,a）：

Q-Learning中的最优动作价值函数表达式：

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\right].$$

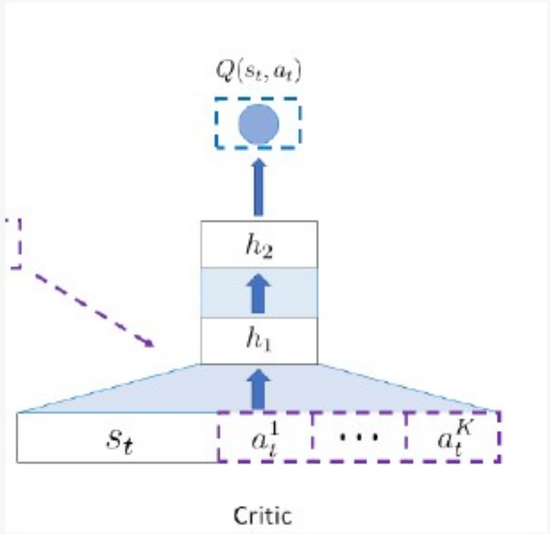
实际使用的推荐系统中，动作价值函数的计算公式：

$$Q(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) | s_t, a_t\right].$$

评价网络的损失函数为：

$$L(\theta^\mu) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\left[(y_t - Q(s_t, a_t; \theta^\mu))^2\right],$$

$$y_t = \mathbb{E}_{s_{t+1}}[r_t + \gamma Q'(s_{t+1}, a_{t+1}; \theta^{\mu'}) | s_t, a_t]$$



2.4The Critic Framework

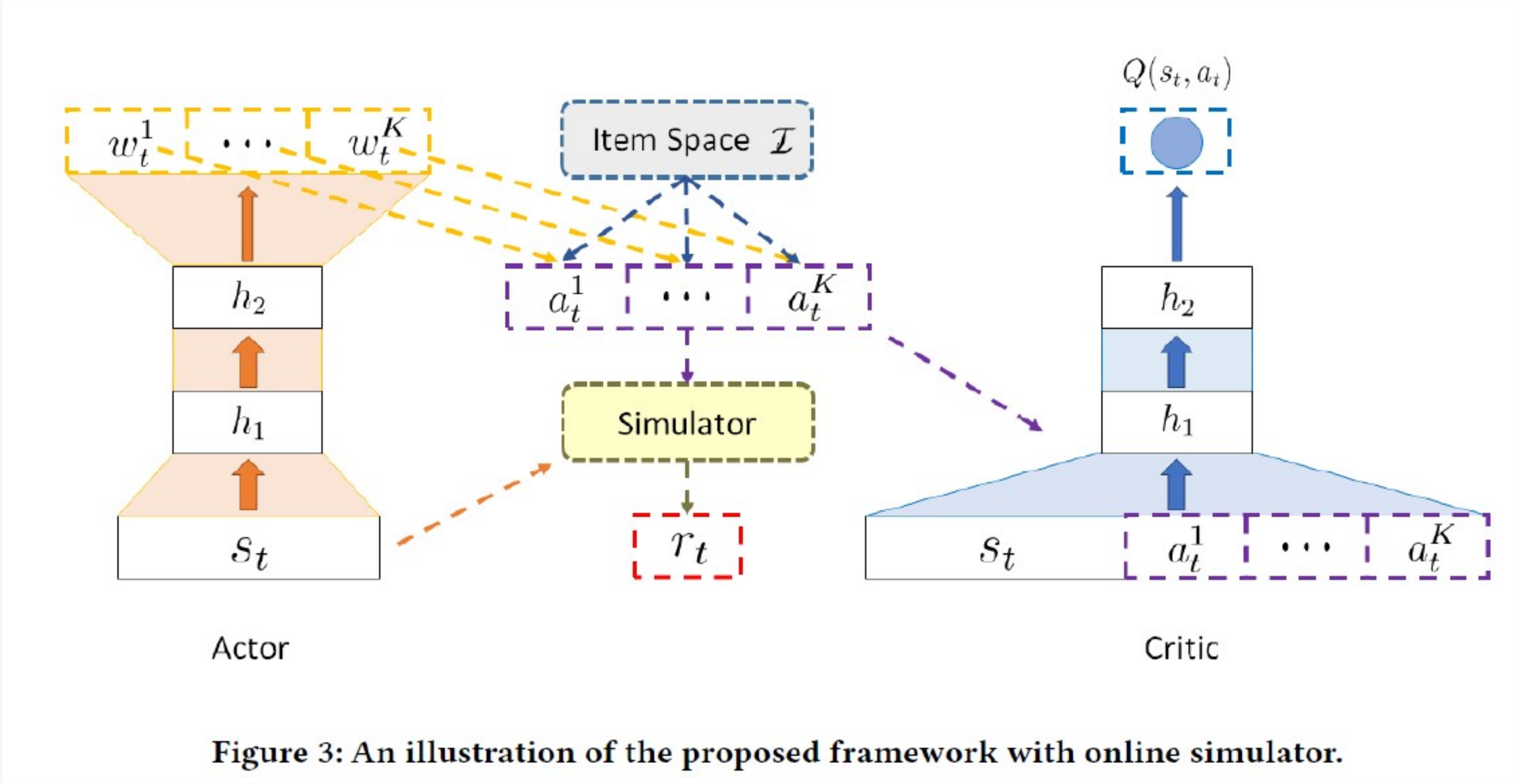


Figure 3: An illustration of the proposed framework with online simulator.

2.4The Critic Framework

在DDPG中，分别使用参数为 θ^{μ} 和 θ^{Q} 的深度神经网络来表示确定性策略 $a=\pi(s|\theta^{\mu})$ 和动作值函数 $Q(s,a|\theta^{Q})$。其中，策略网络用来更新策略，对应 AC 框架中的行动者；值网络用来逼近状态动作对的值函数，并提供梯度信息，对应 AC 框架中的评论家。目标函数被定义为带折扣的总回报：

$$J(\theta^{\mu})=E_{\theta^{\mu}}[r_1+\gamma r_2+\gamma^2 r_3+\cdots]$$

通过随机梯度法对目标函数进行端对端的优化（**注意**，目标是提高总回报 J）。Silver等人证明了目标函数关于 θ^{μ} 的梯度等价于Q值函数关于 θ^{μ} 的期望梯度：

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{\mu}}=E_s[\frac{\partial Q(s,a|\theta^{Q})}{\partial \theta^{\mu}}]$$

根据确定性策略 $a=\pi(s|\theta^{\mu})$ 可得：

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{\mu}}=E_s[\frac{\partial Q(s,a|\theta^{Q})}{\partial a}\frac{\partial \pi(s|\theta^{\mu})}{\partial \theta^{\mu}}]$$

沿着**提升** Q 值的方向更新策略网络的参数。

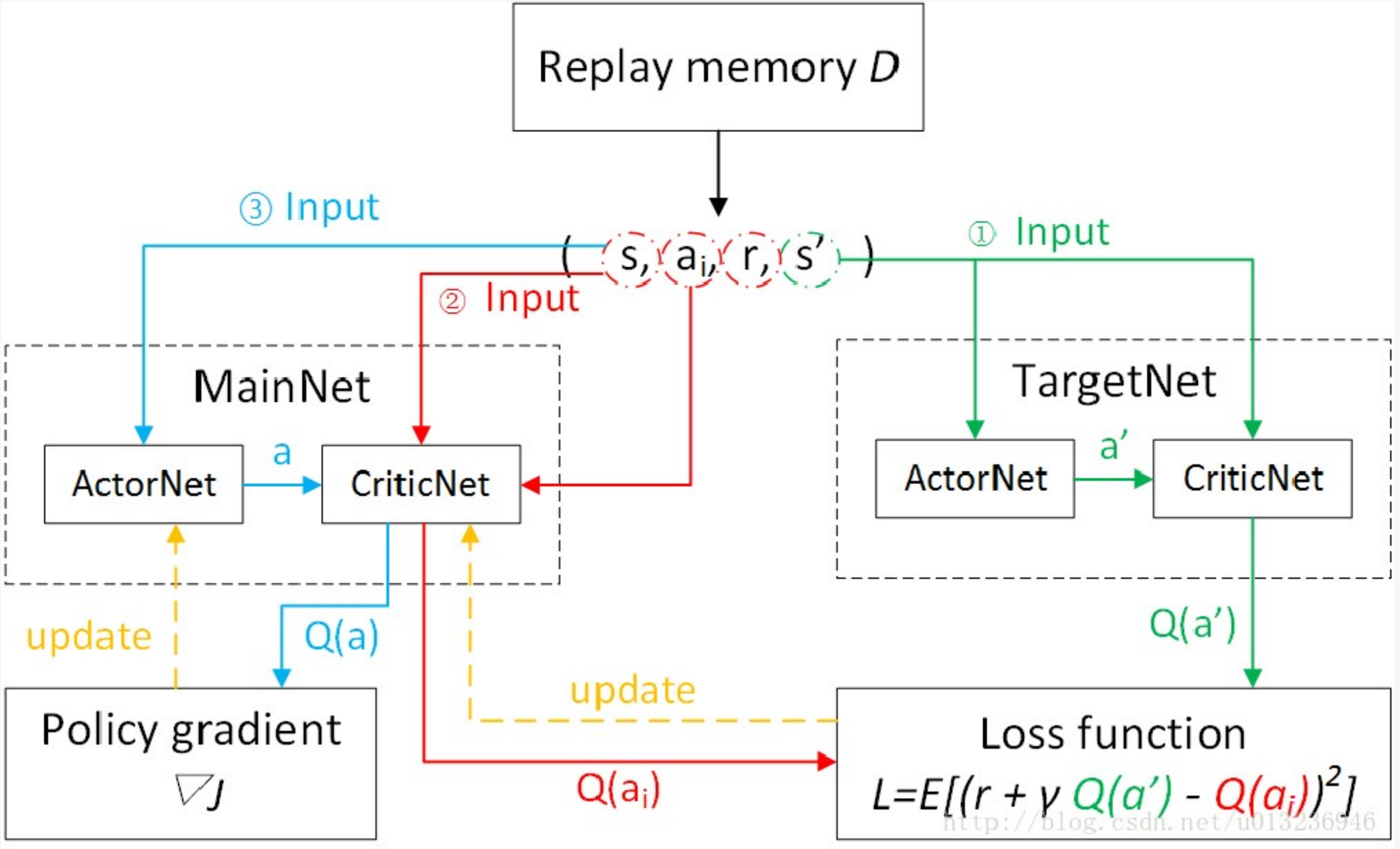
通过 DQN中更新值网络的方法来更新评论家网络，梯度信息为：

$$\frac{\partial L(\theta^{Q})}{\partial \theta^{Q}}=E_{s,a,r,s'\sim D}[(TargetQ-Q(s,a|\theta^{Q}))\frac{\partial Q(s,a|\theta^{Q})}{\partial \theta^{Q}}]$$

$$TargetQ=r+\gamma Q'(s',\pi(s'|\theta^{\mu'})|\theta^{Q'})$$

其中 $\theta^{\mu'}$ 和 $\theta^{Q'}$ 分别表示目标策略网络和目标值网络的参数，用**梯度下降**方式更新值网络。

2.4The Critic Framework



2.4The Critic Framework

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

2.5The Training Procedure

Algorithm 3 Parameters Training for DEV with DDPG.

Initialize actor network f_{θ^π} and critic network $Q(s, a|\theta^\mu)$ with random weights
Initialize target network f' and Q' with weights $\theta^{\pi'} \leftarrow \theta^\pi, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize the capacity of replay memory $\mathcal{D}$
for *session* $= 1, M$ **do**
  Reset the item space $\mathcal{I}$
  Initialize state s_0 from previous sessions
  for $t = 1, T$ **do**
    Stage 1: Transition Generating Stage
    Select an action $a_t = \{a_t^1, \cdots, a_t^K\}$ according **Alg.2**
    Execute action a_t and observe the reward list $\{r_t^1, \cdots, r_t^K\}$ for each item in a_t
    Set $s_{t+1} = s_t$
    for $k = 1, K$ **do**
      if $r_t^k > 0$ **then**
        Add a_t^k to the end of s_{t+1}
        Remove the first item of s_{t+1}
      end if
    end for
    Compute overall reward r_t according Eq. (4)
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Set $s_t = s_{t+1}$
    Stage 2: Parameter Updating Stage
    Sample minibatch of $\mathcal{N}$ transitions (s, a, r, s') from $\mathcal{D}$
    Generate a' by target Actor network according **Alg.2**
    Set $y = r + \gamma Q'(s', a'; \theta^{\mu'})$
    Update Critic by minimizing $(y - Q(s, a; \theta^\mu))^2$ according to:

$$\nabla_{\theta^\mu} L(\theta^\mu) \approx \frac{1}{\mathcal{N}} \left[(y - Q(s, a; \theta^\mu)) \nabla_{\theta^\mu} Q(s, a; \theta^\mu) \right]$$

    Update the Actor using the sampled policy gradient:

$$\nabla_{\theta^\pi} f_{\theta^\pi} \approx \frac{1}{\mathcal{N}} \sum_i \nabla_a Q(s, a|\theta^\mu) \nabla_{\theta^\pi} f_{\theta^\pi}(s)$$

    Update the Critic target networks:

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1 - \tau)\theta^{\mu'}$$

    Update the Actor target networks:

$$\theta^{\pi'} \leftarrow \tau\theta^\pi + (1 - \tau)\theta^{\pi'}$$

  end for
end for

2.6The Testing Procedure

- 训练完成之后，RA可以得到模型的参数
- 因此可以通过模拟的环境进行测试
- 依然通过算法三，与此同时只主要的不同是训练是根据每一个推荐的session，可以通过人工控制session的长度来获取不同长度下推荐系统的表现

实验分析
Experiments

PART THREE

- 3.1Experimental Settings
- 3.2Performance Comparison for Item Recommendations
- 3.3Performance of List-Wise Recommendations
- 3.4Performance of Simulator

Experiment

(1)论文中的框架与其他框架进行对比

(2)list-wise策略对于性能的贡献有多大

3.1Experimental Settings

数据

2017年7月真实的电子商务数据
随机选取100000推荐session
（包含1156675个商品订单）
选取70%作为训练集，30%作为
验证集

RA设置

N=10
K=4

回报设置

滑动/点击/购买，
分别为0,1,5

折扣因子

r=0.75
.

3.2Performance Comparison for Item Recommendations

- CF协同过滤：根据用户的兴趣和选择的偏好，给用户推荐相似的物品
- FM因子分解机：结合支持向量机和因子分解模型，通过矩阵参数建模
- DNN深度神经网络：输入用户历史点击购买的物品，通过Embedding输出下一个推荐的物品
- RNN循环神经网络：通过RNN神经网络根据点击和购买历史记录预测用户接下来购买的物品
- DQN深度Q网络：将用户历史点击购买的物品以及推荐给用户的物品（action）做Embedding，之后输入到神经网络中，通过最优状态价值函数，使用相同的算法进行训练。

3.3Performance of List-Wise Recommendations

为了验证list-wise策略的有效性：

通过改变List-item中K的长度，调整优化LIRD的性能

并与K=1这种特殊情况对比，证明了list-wise的优势

3.4Performance of Simulator

在线模拟器有关键的参数α，可以控制state和action之间的相似性计算

为了学习得到这个参数，论文中通过改变α的大小在较多的物品推荐中，得到合适的参数，通过也要调整其他参数。

PART FOUR

相关工作

4.Related Work

4 Related Work

Chart Name

90 %

Chart Name

68 %

Chart Name

55 %

1.基本的推荐系统知识和模型技术

2.推荐系统的强化学习

将推荐的过程作为用户和推荐系统的交互序列，使得推荐算法从回报中自动得到推荐策略，强化学习所应用的领域以及相关的技术

PART FIVE

总结分析

Conclusion

Conclusion

本文提出了一种LIRD框架，
将推荐缓存作为马尔科夫决策过程，
并使用深度强化学习自动学到推荐策略。
基于推荐系统的强化学习有两种有点：
（1）能够通过与环境进行交互，自动的调整自己的策略
（2）能够从用户的反馈中最大化长期累积回报
文中提出的强化学习框架能够用于大规模电商数据中，并有效地减少时间复杂度

Conclusion

优点

而且设计了在线的环境模拟器，能够对于离线的参数进行预训练和评估，最终将我们的模型与已有的模型进行对比，使用真实的电子商务数据改进了我们模型的性能，并用实验证明list-wise效果比item-wise更好

缺点

缺点：如何对list中的物品进行排序
有效地验证更多的人机交互模式，
如何对商品购物车中的物品进行建模和推荐，
如何更好地设计回报，模拟正负样本作为跳过的信号。

结束语

基于强化学习的推荐系统

THANK YOU!