Learning to Collaborate: Multi-Scenario Ranking via Multi-Agent Reinforcement Learning

Learning to Collaborate——主搜索与店铺内搜索联合优化的初步探索与尝试

作者：冯珺，李恒，黄民烈，刘士琛，欧文武，王志荣，朱小燕

背景与简介：Does your optimization really optimize your application？

在淘宝平台上有非常多的子场景，例如搜索、推荐、广告。每个子场景又有非常多细分，例如搜索包括默认排序、店铺内搜索、店铺搜索等；推荐内有猜你喜欢、今日推荐、每日好店等。基于数据驱动的机器学习和优化技术目前大量的应用于这些场景中，并已经取得了不错的效果——在单场景内的 A/B 测试上，点击率、转化率、成交额、单价都能看到显著提升。 然而，目前各个场景之间是完全独立优化的，这样会带来几点比较严重的问题：

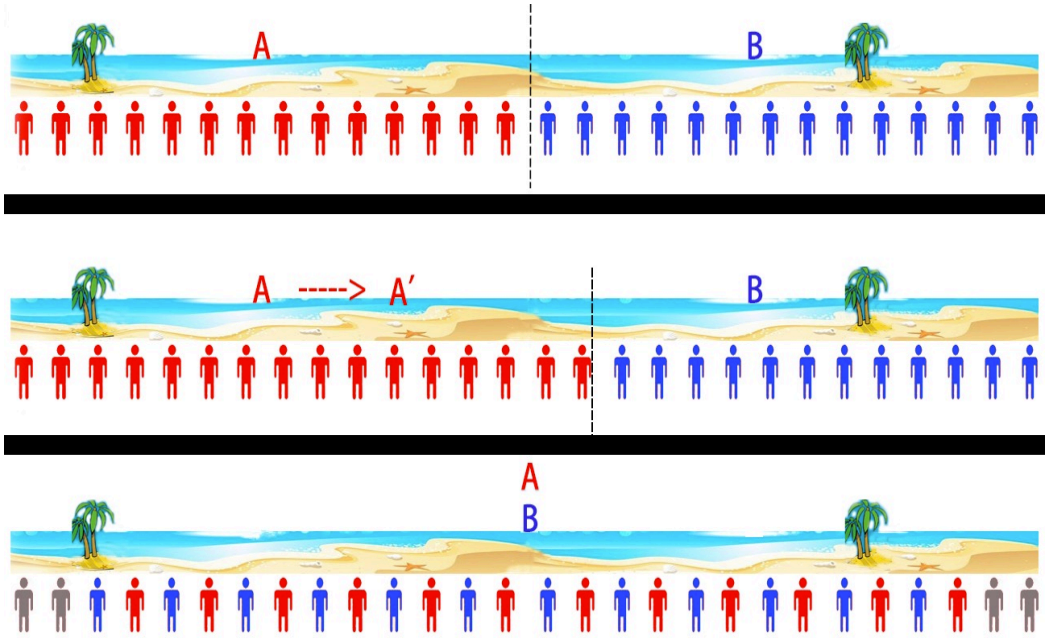
a. 用户在淘宝上购物会经常在多个场景之间切换，例如：从主搜索到猜你喜欢，从猜你喜欢到店铺内。不同场景的商品排序仅考虑自身，会导致**用户的购物体验是不连贯或者雷同的**。例如：从冰箱的详情页进入店铺，却展示手机；各个场景都展现趋同，都包含太多的 U2I（点击或成交过的商品）。

b. 多场景之间是博弈（竞争）关系，**期望每个场景的提升带来整体提升这一点是无法保证的**。很有可能一个场景的提升会导致其他场景的下降，更可怕的是某个场景带来的提升甚至小于其他场景更大的下降。这并非是不可能的，那么这种情况下，单场景的 A/B 测试就显得没那么有意义，单场景的优化也会存在明显的问题。因为这一点尤为重要，因此我们举一个更简单易懂的例子（如下图）。一个 1000 米长的沙滩上有 2 个饮料摊 A 和 B，沙滩上均分分布者很多游客，他们一般会找更近的饮料摊去买饮料。最开始 A 和 B 分别在沙滩 250 米和 750 米的位置，此时沙滩左边的人会去 A 买，右边的人去 B 买。然后 A 发现，自己往右边移动的时候，会有更多的用户（A/B 测试的结论），因此 A 会右移，同样 B 会左移。A 和 B 各自‘优化’下去，最后会都在沙滩中间的位置，从博弈论的角度，到了一个均衡点。然而，最后‘优化’得到的位置是不如初始位置的，因为会有很多游客会因为太远而放弃买饮料。这种情况下，2 个饮料摊各自优化的结果反而是不如不优化的。

多场景问题实际并不止存在于淘宝上，目前比较大型的平台或者无线 APP 都不止一个场景。即使不谈 Yahoo，Sina 等综合性网站，像 Baidu、Google 等功能比较单一、集中的应用，也会有若干场景（如网页、咨询、地图等）。那么这些平台或应用都会面临类似的

问题。 综上，研究大型在线平台上的多子场景联合优化，无论从淘宝平台的应用上，还是从科研的角度，都具有重要意义。

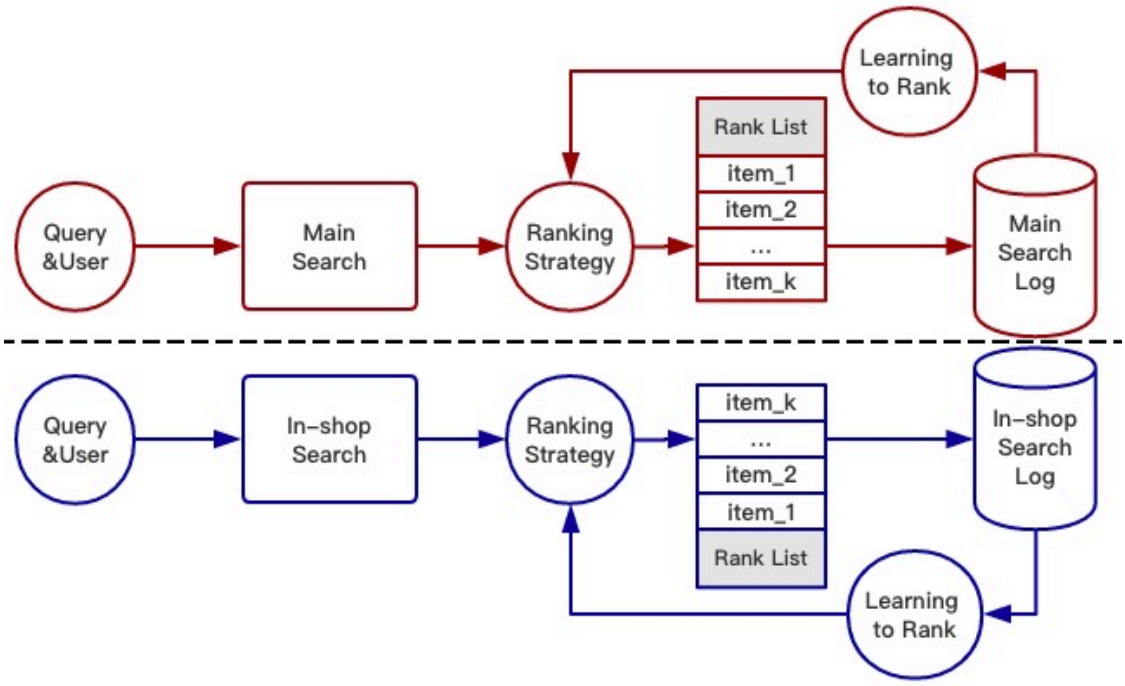
为了解决上述的问题，本文提出一个多场景联合排序算法，旨在提升整体指标。我们将多场景的排序问题看成一个完全合作的、部分可观测的多智能体序列决策问题，利用 Multi-Agent Reinforcement Learning 的方法来尝试着对问题进行建模。该模型以各个场景为 Agent，让各个场景不同的排序策略共享同一个目标，同时在一个场景的排序结果会考虑该用户在其他场景的行为和反馈。这样使得各个场景的排序策略由独立转变为合作与共赢。由于我们想要使用用户在所有场景的行为，而 DRQN 中的 RNN 网络可以记住历史信息，同时利用 DPG 对连续状态与连续动作空间进行探索，因此我们算法取名 MA-RDPG(Multi-Agent Recurrent Deterministic Policy Gradient)。



系统总览

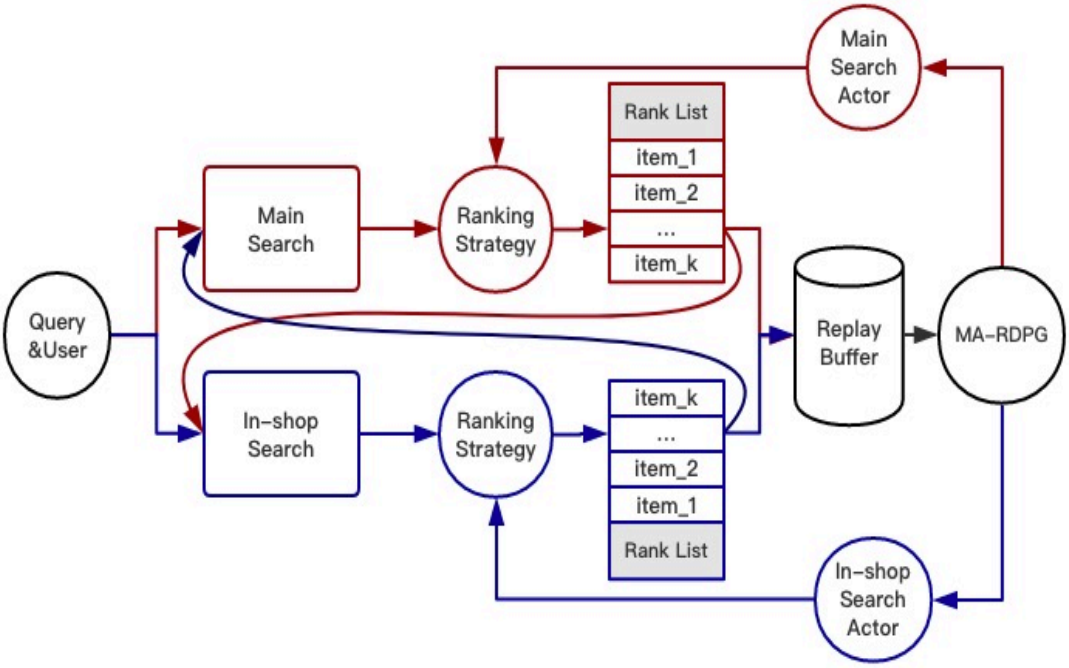
传统的单场景优化

目前，单场景排序策略的大体结构如下，每个商品用一组特征来表示<人气分，ctr 分……>，排序策略通过给出一组特征权重来决定排序的结果，商品的分数即为各个特征的加权相加。主搜索和店铺内搜索都有自己的排序策略，独立优化，互不影响。



多场景联合优化

多场景联合排序的系统结构如下，场景之间不再是孤立的，而是由一个统一算法进行学习，然后将排序模型更新到各个排序场景（Agent）。所利用的用户日志也不再是单场景内部的，而是基于多个场景的信息，其中不仅包含用户的行为反馈，还包括当时所在场景的信息。



方法与应用

multi-agent reinforcement learning

问题描述

为了解决上述单场景优化的两个缺陷，我们将多场景联合优化转化成一个 Multi-Agent Reinforcement Learning 的问题。对问题作如下具体定义：
Multi-Agent: 在多场景环境下，每一个场景的排序策略作为一个 agent，agent 学习自己的 policy function，来将一个 state 映射为一个具体的 action。
Sequential Decision：用户一次只与一个 agent 进行交互，当前 agent 的 action 会影响后面其他 agent 的策略。
Fully Cooperative：所有的 agent 完全合作，来提升一个共同的指标。agent 之间可以互相通信，并且所有 agent 的行为是由一个 centralized critic 进行评价。
Partially Observable： 环境是部分可观测的，每一个 agent 只能看到局部的信息，而观察不到完整的信息。

模型

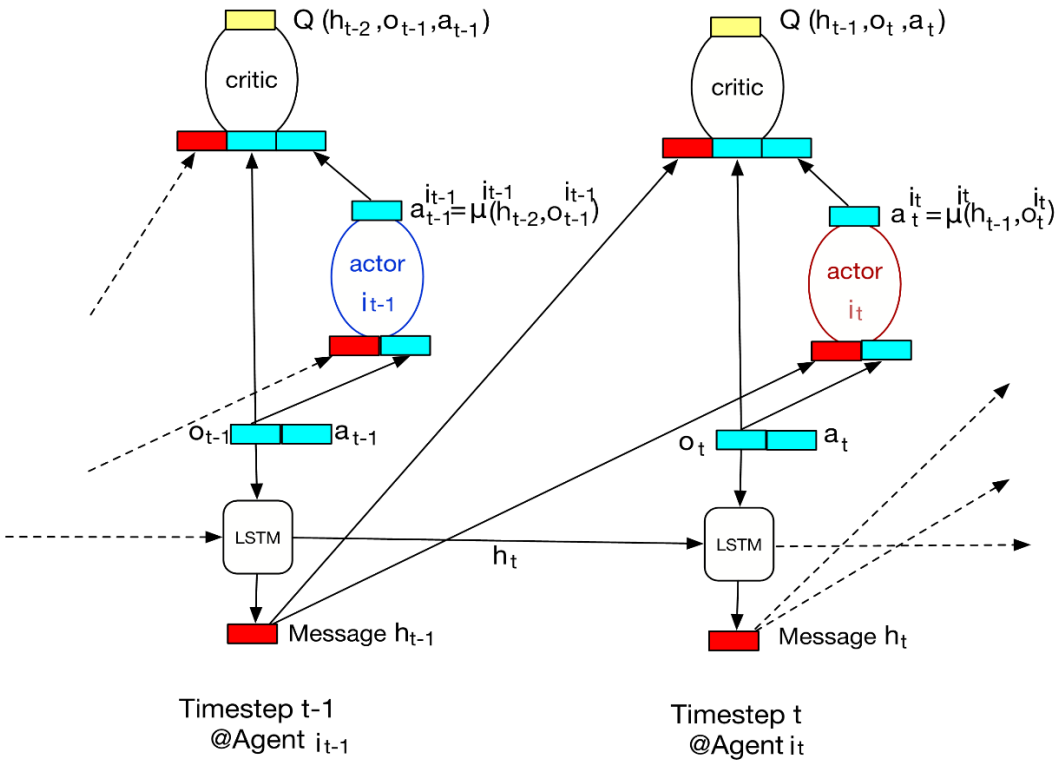
受 DDPG 的启发，我们的模型也是基于 actor-critic 的方法，主要包含三个模块：centralized critic，private actors 和 communication component。centralized critic 近似一个状态动作值函数，表示在当前状态 s 下，执行动作 a 所能获得的未来收益的期望。每一个 agent 的 policy function 由一个 actor network 来表示，输入为当前的状态 s，输出为一个确定的 action。通信模块将之前所有 agent 的 local observation 和 action 编码成一个 message 向量，agent 通过接收 message 来进行合作。通过这些 message 的传递，每个 agent 的决策不仅由自己决定，还会受到其他 agent 的影响。而且 message 还可以帮助 agent 得到更多关于环境信息的描述，帮助 agent 的决策。

我们用$(o_1, r_1, a_1, \cdots, a_{t-1}, o_t, r_t)$ 表示 experience 的序列，其中 o/r/a 分别对应于

observation/reward/action。如前所述，我们的环境是一个部分可观测的，所以状态 St

是前面所有观测的函数，即 $s_t=f(o_1, r_1, a_1, \cdots, a_{t-1}, o_t, r_t)$。N 个 agent $\{A_1, A_2, \ldots, A_N\}$ 分别对应于 N 个不同的优化场景。每个 agent A_i 使用自己的 policy function $\mu^i(s_t)$ 产生 action a_t^i，得到 reward $r_t^i=r(s_t, a_t^i)$，同时状态从 s_t 跳转到 s_{t+1}。当给出一组 action $(a_t^1, a_t^2, \cdots, a_t^N)$，我们使用一个 centralized action-value function $Q(s_t, a_t^1, a_t^2, \cdots, a_t^N)$来对其进行评价。

模型的具体结构见下图所示：



Centralized Critic： 和 DDPG 一样，我们使用一个 critic network 来评估未来的总收益。由于我们的 agent 都共享一个目标，所以我们采用了一个 centralized critic $Q(s_t, a_t^1, a_t^2, \cdots, a_t^N)$。

Private Actor： 每个 agent 有自己的 private actor，来输出自己所要采取的 action。由于我们所要探索的是一个高维连续动作空间，因此我们使用神经网络把 actor 设计成一个确定性的策略，即将 state 直接映射到一个具体的 action。在时间 t，agent A^{i_t}的 action 为

$$a_t^{i_t}=\mu^{i_t}(s_t;\theta^{i_t})\approx \mu^{i_t}(h_{t-1}, o_t^{i_t};\theta^{i_t})$$

，其中$\mu^i(s_t;\theta^i)$是 policy function，θ^i是参数。actor network 的输入 state 有两部分构成：1.当前的 local observation；2.由所有 agent 的历史 observation 和 action 通过 LSTM 编码成的 message。

Communication Component： 将所有 agent 的历史 observation 和 action 编码成 message 向量，

$$h_{t-1}=LSTM(h_{t-2},[o_{t-1};a_{t-1}];\psi)$$

其中 o_t 和 a_t 表示 t 时刻，所有 agent 的 observation 和 action。通过 message 的传递，使得任何一个 agent 都能够得到更完整的信息，有助于 agent 之间更好的合作。

训练

Critic： critic network 使用 Bellman 公式进行学习，最小化如下 loss：

$$L(\phi)=\mathbb{E}_{h_{t-1},o_t,a_t,r_t}[(Q(h_{t-1}, o_t, a_t;\phi)-y_t)^2]$$

其中，$y_t=r_t+\gamma Q(h_t,o_{t+1},\mu^{i_{t+1}}(h_t,o_{t+1});\phi)$

Actor： actor 的更新是最大化 performance function。以 A^{i_t}在时间 t 为例，目标函数如下：

$$J(\theta^{i_t})=\mathbb{E}_{h_{t-1},o_t}[Q(h_{t-1},o_t,a;\phi)|_{a=\mu^{i_t}(h_{t-1},o_t;\theta^{i_t})}]$$

根据链式求导法则，actor 的网络参数更新为：

$$\nabla_{\theta^{i_t}}J(\theta^{i_t})\approx \mathbb{E}_{h_{t-1},o_t}[\nabla_{\theta^{i_t}}Q^{i_t}(h_{t-1},o_t,a;\phi)|_{a=\mu^{i_t}(h_{t-1},o_t;\theta^{i_t})}]=\mathbb{E}_{h_{t-1},o_t}[\nabla_{\theta^{i_t}}Q^{i_t}(h_{t-1},o_t,a;\phi)|_{a=\mu^{i_t}(h_{t-1},o_t)}\nabla_{\theta^{i_t}}\mu^{i_t}(h_{t-1},o_t;\theta^{i_t})]$$

Communication Component： 通信模块实质上为 Critic 和 Actor 的共享部分，更新上面两个模块的时候就可以更新此模块的参数。

算法描述如下：

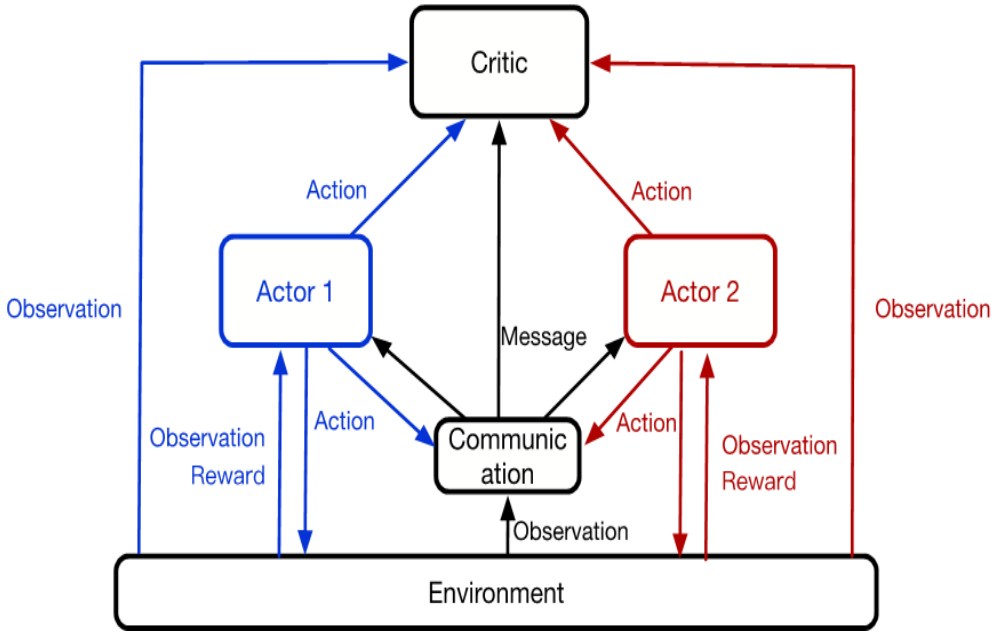
ALGORITHM 1: MA-RDPG

Initialize the parameters $\theta = \{\theta^1, \ldots, \theta^N\}$ for the N actor networks and ϕ for the centralized critic network.
Initialized the replay buffer R
for *each training step e* **do**
  for $i = 1$ *to* M **do**
    h_0 = initial message, $t = 1$
    while $t < T$ *and* $o_t \neq terminal$ **do**
      Select the action $a_t = \mu^{i_t}(h_{t-1}, o_t)$ for the active agent i_t
      Receive reward r_t and the new observation o_{t+1}
      Generate the message $h_t = LSTM(h_{t-1}, [o_t; a_t])$
      $t = t + 1$
    end
    Store episode $\{h_0, o_1, a_1, r_1, h_1, o_2, r_2, h3, o3, \ldots\}$ in R
  end
  Sample a random minibatch of episodes B from replay buffer R
  foreach *episode in B* **do**
    for $t = T$ *downto* 1 **do**
      Update the critic by minimizing the loss:
      $L(\phi) = \mathbb{E}_{h_{t-1}, o_t, a_t, r_t}[(Q(h_{t-1}, o_t, a_t; \phi) - y_t)^2]$,
      where $y_t = r_t + \gamma Q(h_t, o_{t+1}, \mu^{i_{t+1}}(h_t, o_{t+1}); \phi)$
      Update the i_t-th actor by maximizing the critic:
      $J(\theta^{i_t}) =$
      $\mathbb{E}_{h_{t-1}, o_t}[\nabla_{\theta^{i_t}} Q(h_{t-1}, o_t, a; \phi)|_{a=\mu^{i_t}(h_{t-1}, o_t; \theta^{i_t})}]$
      Update the communication component.
    end
  end
end

主搜索与店铺内搜索的应用

淘宝主搜索是淘宝流量的门户，承载着每秒数以万计的查询，为用户提供满意的商品结果。店铺内搜索（无 query 的情况下就是店铺内推荐）也是用户在店铺内淘选商品的重要工具。长期以来两者作为独立的场景，各自优化，都取得了非常不错的效果。由于主搜和店铺内是统一分桶的，所以在日常优化中，我们很容易会发现两者还是存在着一些互相影响。利用 MARL 的方法，将两个场景的排序策略看作两个 agent，同时控制两个场景的排序，优化整体去重的 GMV 指标，这个想法应运而生。下面介绍我们利用 MA-RDPG 算法在主搜和店铺内搜索展开的应用。

首先，应用 MA-RDPG 算法，进行一个 MARL 的建模如下图所示：



Environment： 以主搜和店铺内搜索这两个场景为环境。随着两个 agent 采取 action，环境的状态也发生着改变。并且该环境会跟据所采取的 action 和所处的 state，给出一个 reward。

Agents： 主搜索和店铺内搜索分别是两个 agent。对同一个用户而言，在同一时刻，只被一个 agent 所服务。

States： 我们的 state 包含两个方面，一个是 communication 模块所产生的上一个时刻的 agent 所发出的 message，另一个是当前 agent 所观测到的 local observation。local observation 包括：1. 用户的静态属性信息（性别、年龄和购买力等）；2. 用户点击商品的属性信息（价格、CVR 和销量等）；3. query 的行业类型、以及当前用户所在的页码编号；4. 当前所处场景的 index。10 维的 message 与 52 维的 observation concat

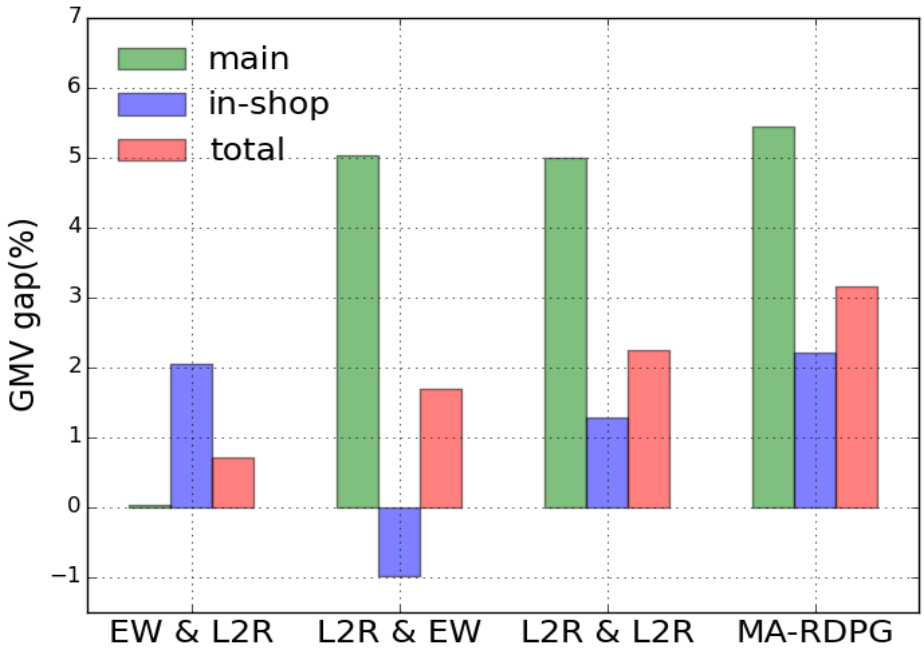
起来形成 state 向量。

Actions： 如前所述，目前线上的排序是根据商品特征线性加权求和所得的分数，来进行排序的。如果控制了这些排序特征的权重，即控制了排序结果。主搜 action 为一个 7 维权重向量，店铺内搜索 action 为一个 3 维权重向量。

Reward： 如果用户在此次 pv 上发生了点击行为，则 reward 为 1。若用户发生了购买行为，则购买金额为 reward，若既无点击也无成交，则 reward 为-1。

实验与分析

在线训练过程如本文**系统总览**中的图所示。actor network 是一个 32/32/7(3)的全连层网络，前两层由 ReLu 激活，最后一次用 softmax 激活。critic network 是一个三层的全连层网络，用 ReLu 激活。γ 设置为 0.9。设置 buffer size 为10^4，minibatch size 为 100。我们进行了四组对比实验：1. Empirical Weight（EW，经验权重） + Learning To Rank（L2R，目前线上在使用的 online LTR）；2. L2R + EW；3. L2R + L2R；4. MA-RDPG。A + B 表示主搜部署 A 算法，店铺内搜索部署 B 算法。每组实验都与 EW+EW 作比较，A/B test 结果如下图所示：

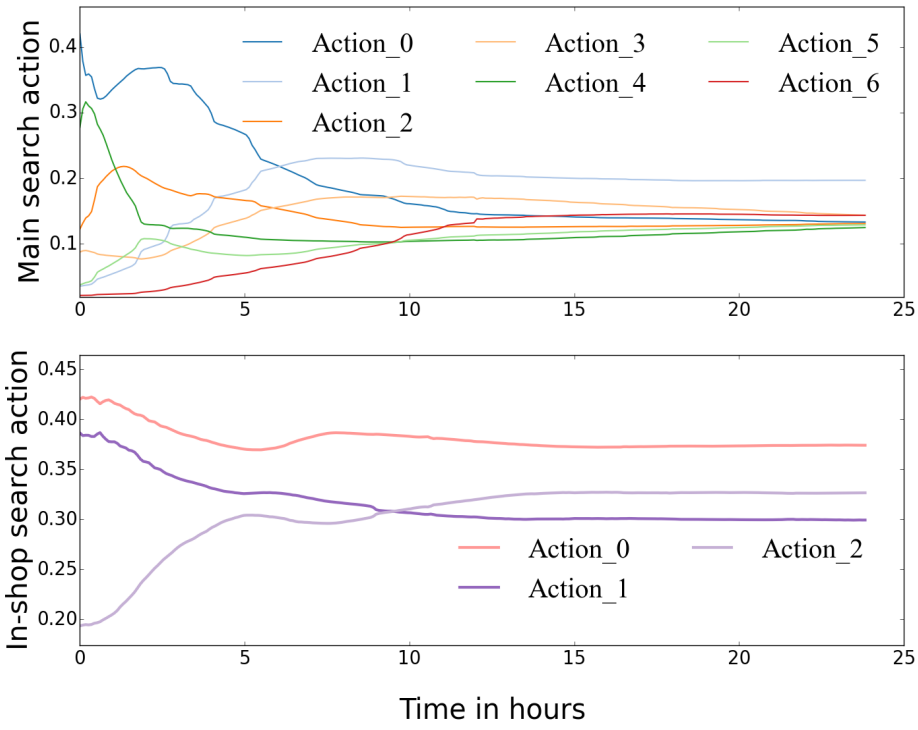


结果分析

首先，MA-RDPG 的效果显然优于其他方法，甚至都不逊于目前线上最好的 L2R+L2R。比起单场景独立的优化，我们的联合优化方法所取得的效果证明了场景之间确实有合作的空间，在多场景的协同配合之下，整体指标确实会得到提升。其次，我们可以看到 MA-RDPG 与 L2R + L2R 相比，在没有损失主搜指标的情况下，提升了店铺内搜索的 GMV，场景之间的协助再一次得到了体现。最后，实验 L2R + EW 也应证了主搜效果的提高，会一定程度上伤害店铺内搜索的 GMV 这个结论。

训练过程

我们监控每个训练 batch 所输出的平均 action，观察它们在训练过程中的变化情况。如下图所示：

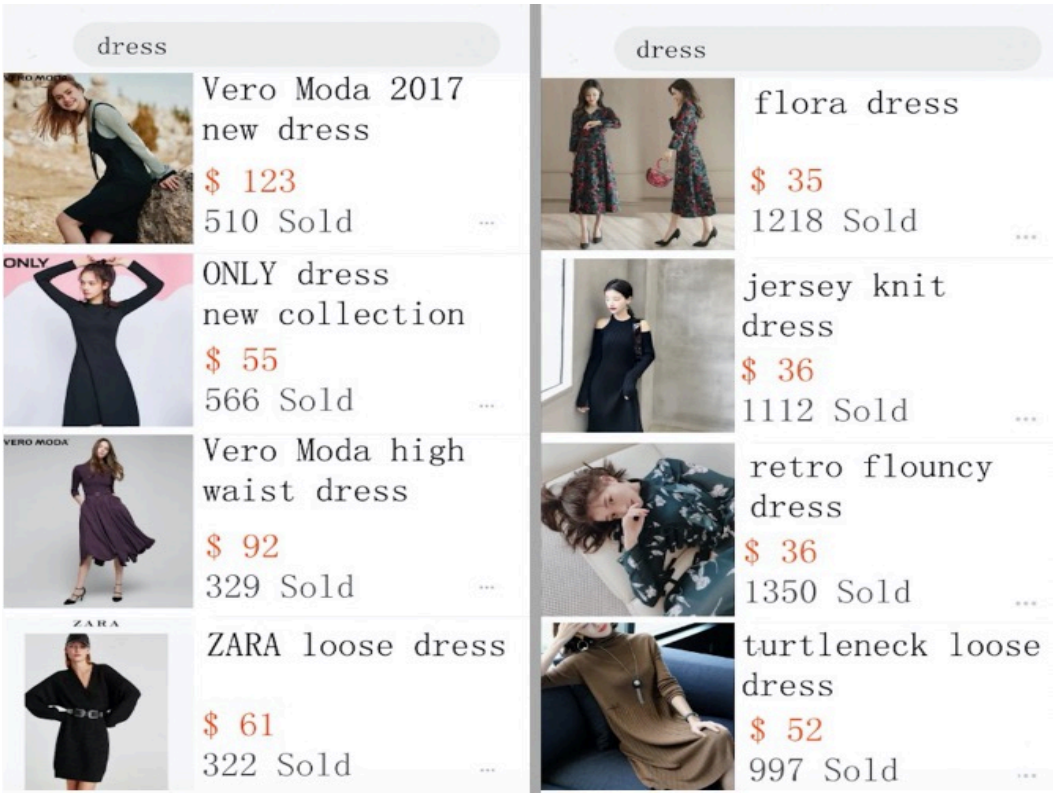


可以看到，actor 所输出的 action 随着训练时间而逐步趋向稳定。上图为主搜的 action，下图为店铺内搜索的 action。输出权重的大小顺序也基本与我们的业务常识相符合。

Case Study

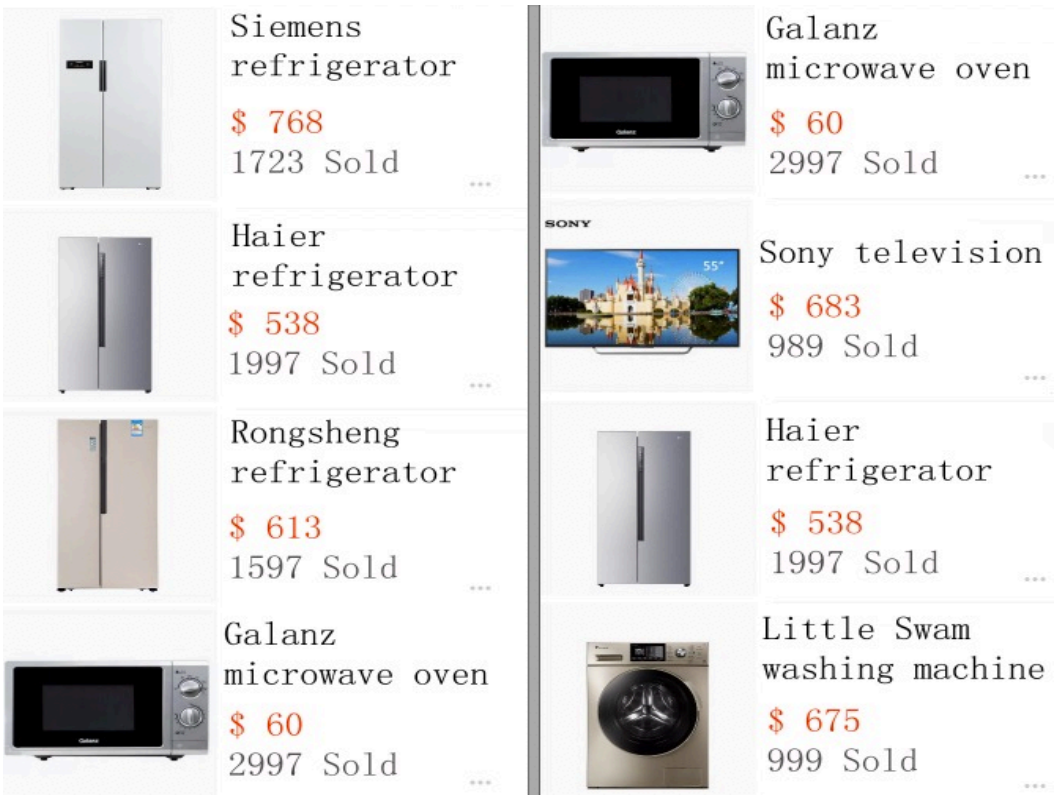
我们选取几种典型的 case 来进一步说明，MA-RDPG 是如何促使两个场景进行合作的。

case 1: 主搜是如何帮助店铺内搜索。模拟这样一个情景，一个高购买力的年轻女性，在寻找价格较高的商品，搜索“dress”，在主搜场景下，对比 MA-RDPG 和 L2R 给出的搜索结果。



左边 MA-RDPG 给出的结果多数是大店和大品牌的商品，吸引用户进店，从大局出发着眼于未来收益。而右边 L2R 还是只考虑本场景内的收益，给出一些高转化与高销量的商品，希望引导用户尽快成交。

case 2: 店铺内搜索利用用户在主搜场景的信息改善排序。一个男性用户想购买一台冰箱，起初在主搜搜索“冰箱”，并点击浏览一些冰箱的商品，之后看到一款大型家电店铺的商品，点击并进入到店铺内搜索页面，比较 MA-RDPG 与 L2R 的店铺内搜索页结果如下：



显然，左边考虑了用户在主搜场景内的信息，将更多的冰箱类目的商品排在前面，更加迎合用户的需求。而右边的结果并不能体现这一点。

实验环境下的结果

主搜和店铺内上线，测试效果如下：

单场景效果：在综合优化下，主搜与基准桶相比，算法相关指标提升+12.5%；店铺内对比基准桶，算法相关指标有+9.4%的提升。

联合效果：在综合优化下，联合去重报表对比基准桶，有近+10%的提升。

总结与展望

随着 AI 技术的发展，越来越多的新技术涌现，我们的排序也经历了从规则到浅层监督学习，到深度学习、强化学习等智能排序算法的转变。新的问题与挑战，迫使我们不断的开拓与进取。从一个场景的优化，到现在尝试着联合两个场景一起优化，这只是联合优化的一小步，现在的做法也比较简单，甚至还存在着非常多的不足和缺陷，还有更多更复杂的问题需要我们去克服去解决。我们相信在面对未来越来越复杂的电商排序场景下，平台需

要的不是各场景之间的互搏与内耗，而是协同与合作，我们期待更多更高级的多场景联合优化的方法涌现，为平台创造更大的价值。