22 强化学习

美国密西根大学教授 Satinder Singh：强化学习中的“发现”

整理人：智源社区 吴继芳

第二届北京智源大会“强化学习”专题论坛上，Satinder Singh 教授做了题为《强化学习中的发现》(Discovery in Reinforcement Learning) 的主题演讲。Satinder Singh，美国密西根大学教授，Deep Mind 首席科学家，AAAI Fellow。

在本次演讲中，Satinder Singh 教授系统地介绍了他与其学生、同事近期关于强化学习的两个研究工作。报告主要讨论如何尝试通过 Meta-Gradient 方法来学习发现以往强化学习智能体中需要手动设置的参数：内在奖励和辅助任务问题。针对于如何通过数据驱动的方式学习到内在奖励函数，他们提出了一个学习跨多生命周期 (lifetime) 内部奖励函数的 Meta-Gradient 框架，同时设计相关实验证明学习到的内在奖励函数能够捕获有用的规律，这些规律有助于强化学习过程中的 exploration 和 exploitation，并且可以迁移到到不同的学习智能体和不同的环境中。针对于如何在数据中发现问题作为辅助任务，他们扩展通用辅助任务架构，参数化表示 General Value Functions，并通过 Meta-Gradient 方法学习更新参数发现问题。实验证明这个方法可以快速发现问题来提高强化学习效果。下面是智源编辑为大家整理的讲座内容。

一、“发现”的含义

什么是强化学习中的“发现”？简单地思考，强化学习智能体中的参数可以分成两部分：一部分参数是从数据中学习发现得到，另一部分是由研究人员根据经验手动设置。Satinder Singh 教授的报告主要讨论他和他的团队如何尝试通过 Meta-Gradient 方法来学习发现参数。在强化学习中，策略 (policy) 函数和价值 (value) 函数的参数值通常从数据中学习得到。对于那些通常手动设置的参数，如图 1 所示，表格中是最新论文中的一些例子以及它们的出处。这些例子都是采用 Meta-Gradient 方法发现参数。有些通过元学习 (Meta-Laring) 发现一个好的策略参数初始值。有些是用 Meta-Gradient 方法发现学习率 (learing rate) 和折扣因子 (discount factor)。有些是用 Meta-Gradient 方法发现内在奖励 (intrinsic rewards) 和辅助任务 (auxiliary tasks) 等。在本次报告中，Satinder Singh 教授主要分享他和他的团队近期发表在 ICML 2020 和 NeurIPS 2019 中的两篇论文的相关研究工作 (图 1 中标红的两篇)。虽然有许多不同的发现方法，比如：基于人口的方法 (population based method)、进化方法 (revolution method)，但是 Satinder Singh 教授他们只是采用启发式搜索方法发现超参数值。这次报告的重点是采用 Meta-Gradient 方法发现参数。

Initial Policy Parameters	Finn et.al. (ICML 2017)
Learning rate, discount factor	Xu et.al. (NeurIPS 2018)
Intrinsic Rewards	Zheng et.al. (NeurIPS 2018), Zheng et.al. (ICML 2020)
Auxiliary Tasks	Veeriah et.al. (NeurIPS 2019); Zahavy et.al. (2020)
Loss fn. parameters	Zahavy et.al. (2020)
Options	Veeriah et.al. (2020)

图 1：手动参数的最新研究方法

二、内在奖励

第一项工作由 Satinder Singh 教授和他的博生生共同完成的。文章的题目是:《What can Learned Intrinsic Rewards Capture ?》[1]。

2.1 研究动机

在强化学习中，智能体有很多结构存储知识。这些结构分为: 常见结构 (common structure) 和非常见结构 (uncommon structure)。其中，常见结构有 : 策略 (policies)、价值函数 (value functions)、环境模型 (models) 和状态表示 (state representations) 等。在本次报告中，主要关注非常见结构: 奖励函数 (reward function)。之所以是非常见结构是因为在强化学习中这些奖励通常都是根据环境决定，并且是不可改变的。在论文中，将强化学习问题中的奖励函数分为外在奖励 (extrinsic rewards) 和内在奖励 (intrinsic rewards)。外在奖励用来衡量智能体的性能，通常是不能改变的。内在奖励是智能体内部的。在内在奖励中，有很多方法用来存储知识，但是这些方法都是手动设计的，比如: reward shaping、novelty-based reward、curiosity-driven reward 等。这些手动的内在奖励方法都依赖领域知识或者需要细致的微调才能起作用。在本次报告中，Satinder Singh 主要关注两个研究问题:

1. 是否能够通过数据驱动的方式，学习得到一个内在奖励函数?
2. 通过学习到的内在奖励函数，什么样的知识能够被捕获到?

针对第一个问题，论文中提出了一个学习跨多生命周期 (lifetime) 内部奖励函数的可扩展的 Meta-Gradient 框架。针对第二个问题，论文中设计了一系列的实验，通过实验证明: 1) 学习到的内在奖励函数能够捕获有用的规律，这些规律有助于强化学习过程中的 exploration 和 exploitation; 2) 学习到的内在奖励函数可以推广到不同的学习智能体和不同的环境中; 3) 内在奖励函数可以捕获知识告诉智能体要做什么而不是怎么做，策略是用来告诉智能体应该怎么做。

2.2 Optimal Reward Framework

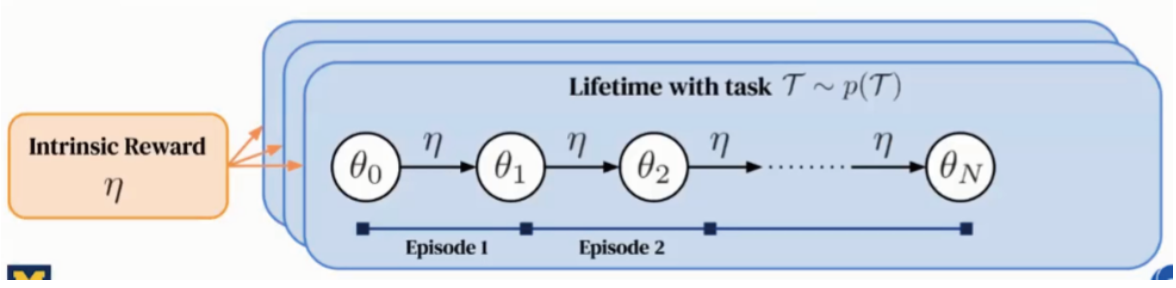


图 2: 基于多生命周期的最优奖励框架

研究的目标是从经验中学习到有用的内在奖励，论文中考虑多生命周期 (multiple lifetime) 的情况 (如图 2 所示)。涉及到的相关概念定义如下:

生命周期 (lifetime): 是指由多个情节 (episodes) 构成的智能体的整个训练时间。生命周期可以有成千上万的情节。每个情节有多个不同的参数更新 (如图 1 所示)。在生命周期的开始，智能体被按照一定分布随机采样的

任务初始化。 在实验过程中，任务可以是静态（stationary）或非静态（non-stationary）的。当情节属于一个生命周期时，任务随着时间改变。

内部奖励（intrinsic reward）：在一个生命周期内被用来训练智能体策略（policy），策略不能看到外在奖励，只能利用内部奖励更新参数。

最优奖励问题（optimal reward problem）：是指学习跨多生命周期的内在奖励函数。通过训练多个随机初始化的策略，获得累积最优的外在奖励。

目前针对最优化内在奖励函数，仍然存在两个开发不足方面的问题：

1. 内在奖励函数的输入应该是整个生命周期的行为历史而不是情节（episode）内的行为历史。这样有助于强化学习中的 exploration。这是因为 exploration 需要关注跨多个 episodes 的发生情况，即当进行探索时，需要观察整个生命周期内的行为历史。
2. 需要最大化整个长生命周期的返回（lifetime return），而不是单个情节的返回（episodic return）。这样可以有更多空间来平衡跨多个场景之间的 exploration 和 exploitation。

针对以上的不足，Satinder Singh 教授和他的团队提出一个 scalable gradient-based 的方法来解决最优化内在奖励的问题。

2.3 Truncated Meta-Gradients with Bootstrapping

Meta-Gradient 方法有两个循环，分别是：内部循环（inner loop）和外部循环（outer loop）。

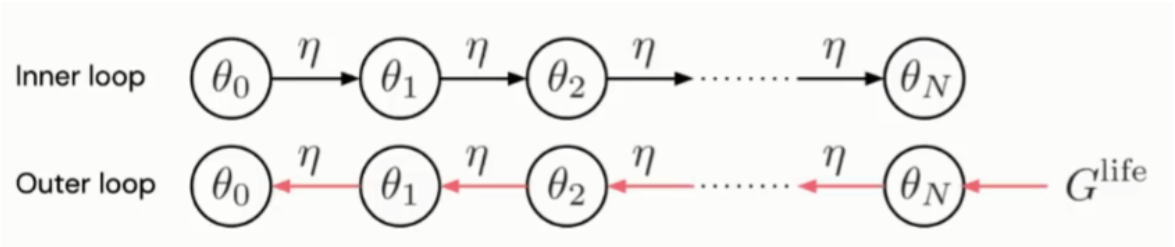


图 3：Meta-Gradients 中的两个循环

内部循环在强化学习中很常见，如图 3 所示，在展开的计算图中进行策略参数（θ）更新，直到生命周期结束。外部循环是指在整个生命周期通过反向传播方法，更新内在奖励函数参数（η）。内在奖励函数可以是一个单独的神经网络或者是循环神经网络（RNN）。神经网络的节点与生命周期中各情节的行为历史对应，构成整个生命周期的行为历史。Meta-Gradient 方法的核心思想是：在内部循环中的策略参数更新是关于内在奖励函数参数可微到。因此，如果我们能够展开计算图，在整个生命周期内更新策略参数（θ），那么就可以通过这个巨大的计算图反向传播计算关于内在奖励函数参数（η）的 meta-gradients。当然，由于受到内存空间的限制，展开整个计算图更新参数是不可行的。这也是为什么之前的一些 meta-gradients 方法在更新了少量θ参数后就调优情节返回（episode return）的原因。为了解决这个挑战，Satinder Singh 教授仍然采用了截断计算图只更新少量参数的方法。但是他们采用生命周期价值函数（lifetime value function）来近似截断内部循环参数更新之后的其他未

展开计算图节点的生命周期奖励（如图 4 所示）。这个生命周期价值函数在未来不仅用来预测整个生命周期外部奖励，同时不断的更新内部循环中的参数θ。因此这个价值函数是在多生命周期间进行训练。

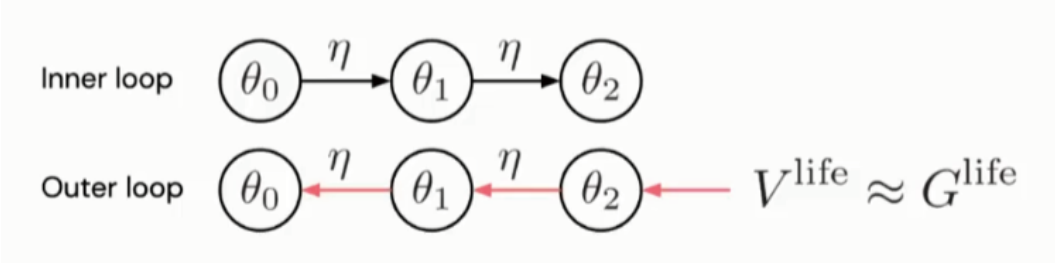


图 4：Truncated Meta-Gradients with Bootstrapping

2.4 实验

Satinder Singh 教授分享了他们是如何实验通过 meta-gradients 方法更新内部奖励函数参数的。

2.4.1 方法（Methodology）

1. 设计特定领域的一系列具有特定规律的任务。
2. 训练跨多生命周期的内部奖励函数。
3. 评价分析学习到的内部奖励函数在新的生命周期的效果。

2.4.2 探索不确定状态

实验设置如图 5 所示，引入四个环境房间，智能体（图中蓝色方块）在每个 episode 中找到不可见的目标位置。这个目标位置不同，生命周期不同，由随机采样得到。但是每个生命周期的目标位置是确定的。如果智能体到达目标位置，则当前 episode 结束。

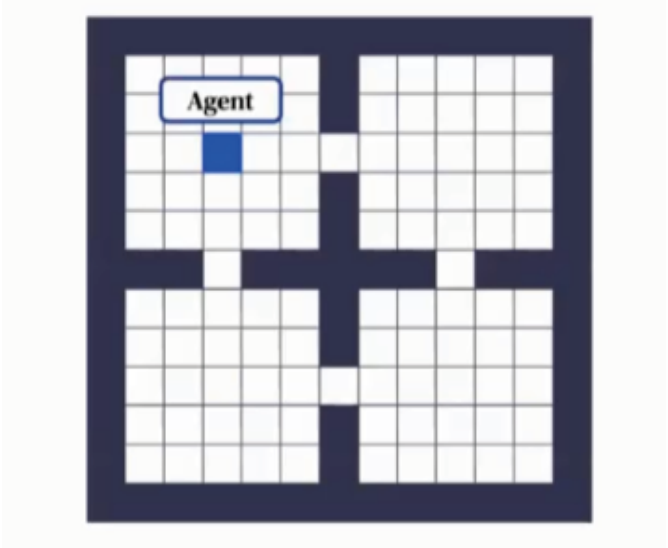


图 5：探索不确定状态实验

这个实验中的最优表现应该是：在生命周期的第一个 episode, 智能体可以高效的探索整个房间来找到目标位置。当找到目标位置，则第一个 episode 结束。从第二个 episode 开始，智能体能够记住目标位置在哪并直接到达目标位置。

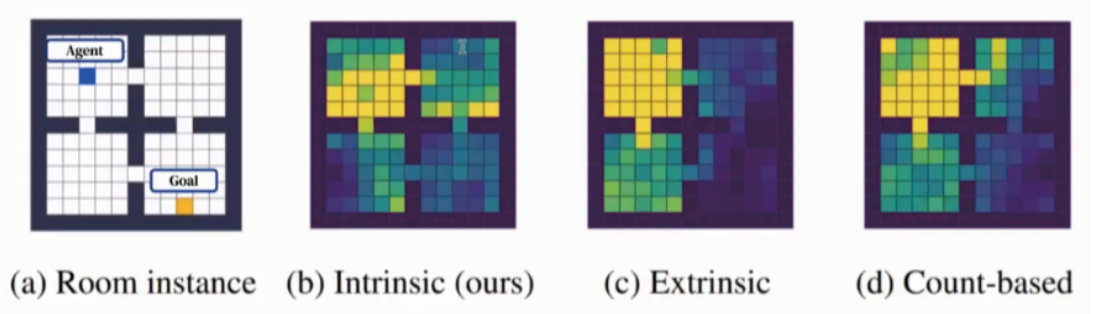


图 6：不同方法的智能体探索轨迹对比分析

图 6 中展示了生命周期内智能体的探索轨迹，图 (a) 中黄色的位置代表对于智能体不可见的目标位置，(b) 是智能体采用学习到的内在奖励函数方法，(c) 是智能体采用外部奖励方法，(d) 是智能体采用 count–based exploration 方法。可以看到，(b) 中有更多的黄色和绿色充满了四个房间，这表明采用学习到的内在奖励函数在探索过程中表现更好。

2.4.3 探索不确定的目的

实验设置如图 7 所示，有三个目的 (object)A、B、C，它们分别具有不同的奖励，A 表示不好也不坏，B 表示总是坏的，C 表示并没有那么好。这些 object 在不同的生命周期内不同，通过随机采样产生，在特定的生命周期内是固定的。智能体收集到这三个目的中的任何一个，则该 episode 结束。

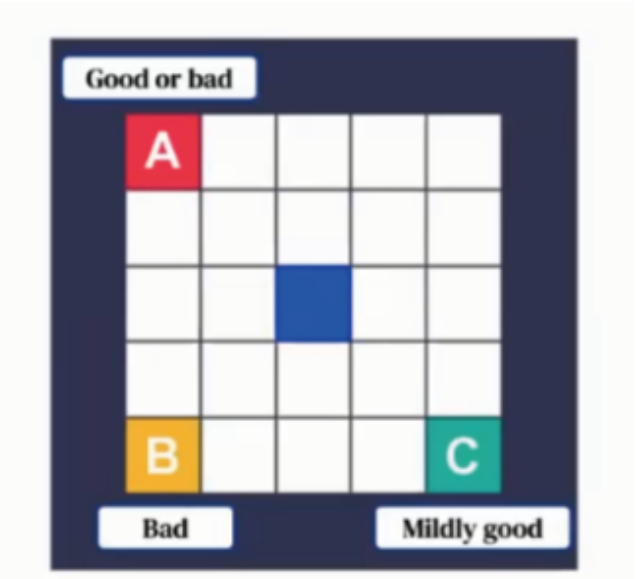


图 7：探索不确定目的实验

在这个实验中，我们希望学习到内在奖励函数可以捕获到的规律是：规避 B，因为它总是不好的，同时可以快速的指出 A 和 C 到底哪个更好，并在剩下的生命周期内总是做出最好的选择。为了学到这些规律，智能体需要跨多个 episode 进行探索学习。图 8 展示了不同的 episode 中探索到的 object，图中每个方格代表每个轨迹中积累的内在奖励，蓝色代表正的奖励，红色代表负的奖励。可以看到在 Episode 1 中，推荐 object A，在 Episode 2 中推荐 object C，在 Episode 3 中推荐 object A，整个过程中都没有推荐 object B，它的颜色总是红的。

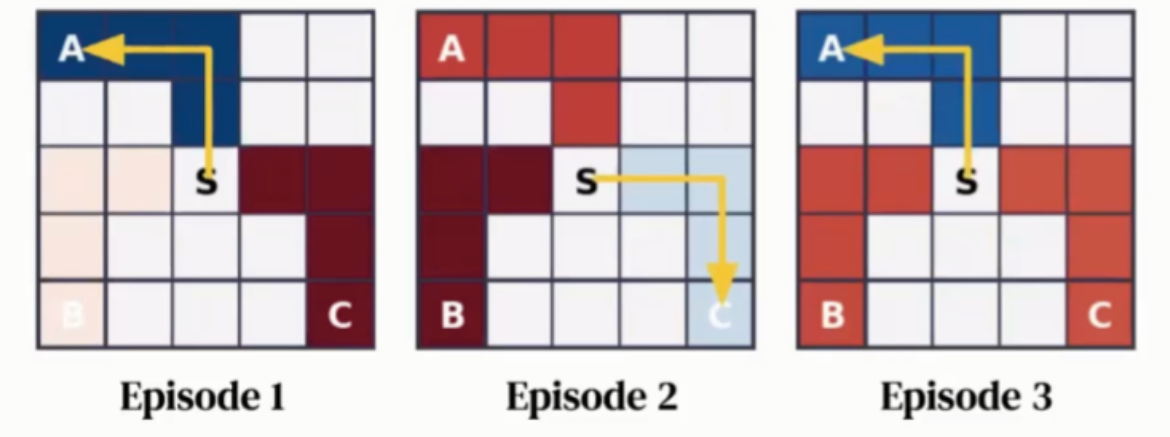


图 8：每个轨迹的内部奖励可视化

2.4.4 处理非固定任务

在这个实验中，假设 A 和 C 的外部奖励在一定时间后会发生变化，智能体需要学习预测这个变化是在什么时候发生，以此来改变策略适应新的任务。图 9 展示了实验结果，左侧图中蓝色柱状条代表内部奖励，在开始时内部奖励一直是正的，大约在 Episode 400 接近 500 的时候，内在奖励开始变为负值（绿色框的部分），即智能体开始缓慢的调整策略，到达 Episode 500 时，策略调整为一个新的行为来应对新的任务。右侧图展示了整个过程策略的 Entropy 变化，可以看到在前 400 个 episodes 中 Entropy 一直保持很小，当内部奖励为负值时，智能体不知道任务会如何改变，因此它的 entropy 开始增加。紧接着，智能体可以快速学习到这些改变，做出策略调整，快速适应新的任务。

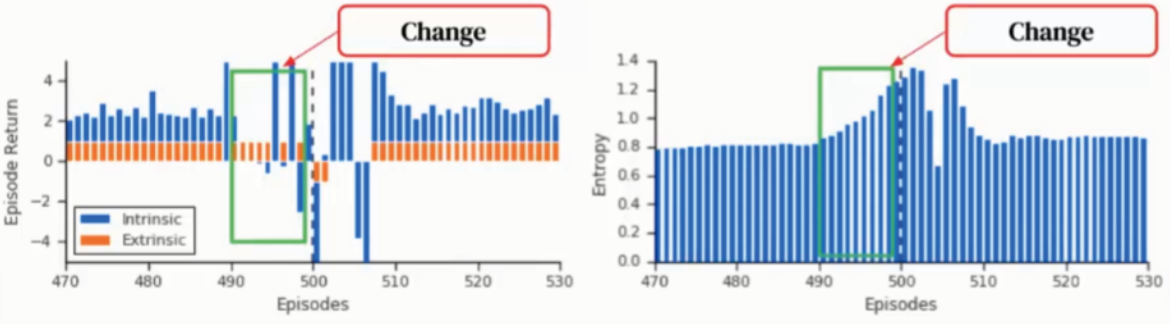


图 9：处理非固定任务实验结果

2.4.5 性能 (performance)

设计实验与手动设置内部奖励方法进行比较，如图 10 所示，前三个子图代表静态任务，在三个子图中学习到的内在奖励函数获得最高的 episode return。最后一个子图代表非静态任务，可以看到在任务发生改变时通过学习到的内在奖励函数智能体的表现可以最快地恢复到最佳状态。

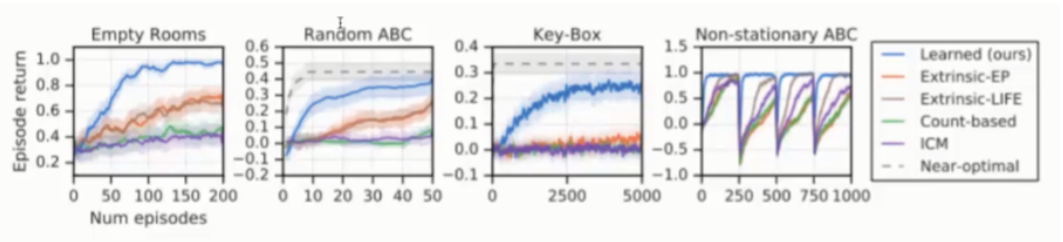


图 10：Learned v.s. Handcrafted Intrinsic Rewards

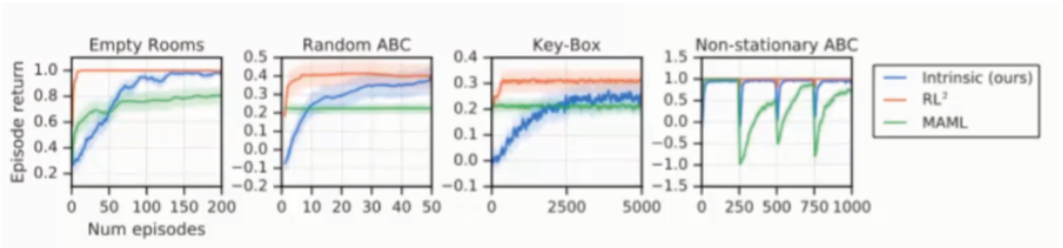


图 11：Comparison to policy transfer methods

图 11 中展示的是与策略转移 (policy transfer) 方法 (比如：MAML、RL2) 的实验比较结果，可以看出内部奖励方法的表现优于 MAML，最终达到同 RL2 一样的效果。这是因为内部奖励方法需要从部分 episode 中学习策略，而 RL2 有一个好的初始化策略。

2.4.6 迁移到新智能体环境

因为有些情况下策略是不能转移的，所以通用转移内部奖励比策略转移更可行。图 12 中，采用新的动作空间 (action space) 来验证训练得到的内部奖励，因此策略无法进行转移。permuted actions 是指左 / 右和上 / 下的语义相反，extended actions 是指添加 4 个对角移动的动作。从图中可以看到学习到的内在奖励可以很好地转移到新的动作空间中，对新环境是敏感的。

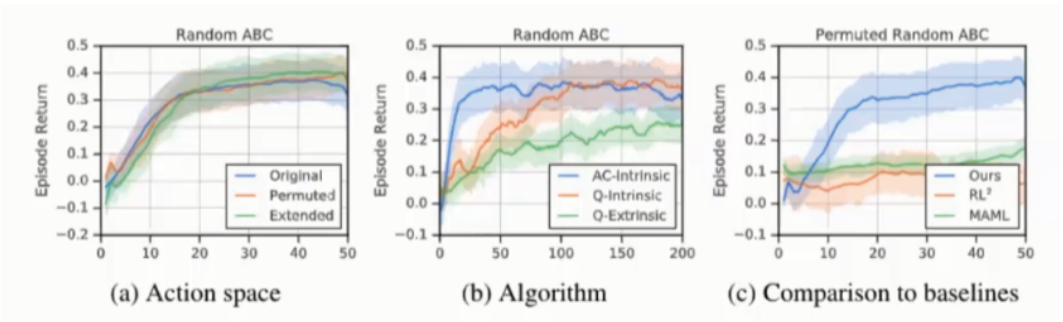


图 12：Generalisation to new agent–environment interfaces in Random ABC

2.4.7 Ablation Study

如图 13 所示，蓝色曲线代表将 lifetime 的历史行为作为输入的 LSTM 内部奖励网络，橙色曲线代表将 Episode 历史行为作为输入的 LSTM 内部奖励网络，绿色带表去掉 lifetime 历史行为。从图中可以看出绿色曲线表现最差。这表明智能体在探索过程中，lifetime history 很重要。橙色曲线基本都比绿色曲线表现差，这也表明了 long-term lifetime history 在智能体平衡 exploration 和 exploitation 过程中是必需的。

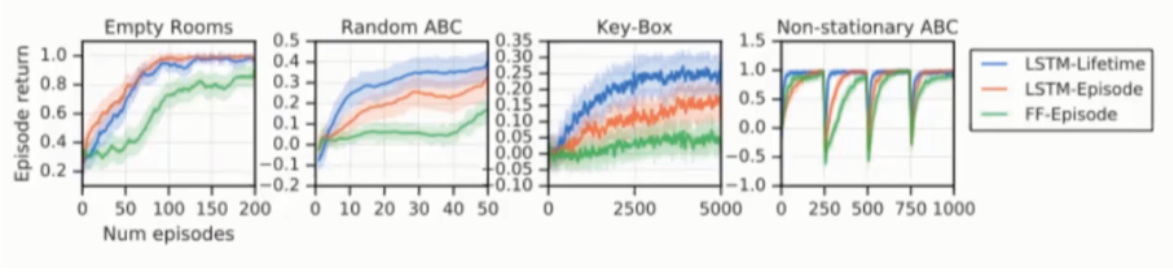


图 13：Evaluation of different intrinsic reward architectures and objectives

2.4.8 总结

文章中证明通过 Meta-Gradient 方法可以学习到有用的内在奖励。学习内在奖励可以捕获到有用的规律应用于智能体的 exploration 和 exploitation。同时捕获到的知识可以迁移到其他学习环境的智能体上。目前该方法仍然太简单，有很多限制，Satinder Singh 教授他们未来将研究在更加复杂环境下的内在奖励学习。

三、辅助任务

第二项工作由 Satinder Singh 教授和他的 DeepMind 同事共同完成的。文章的题目是:《Discovery of Useful Questions as Auxiliary Tasks》[2]。

3.1 预测问题

基本上所有的机器学习研究都是通过学习回答预先定义好的问题。为了能够实现更一般的人工智能，智能体需要能够自己发现问题并回答这些问题。在本文中，作者关注于将发现问题作为辅助任务来帮助构造智能体的表示。

3.2 General Value Functions (GVFs)

General value Functions 是指表示任意状态特征的价值函数，是强化学习中的价值函数的扩展，它可以由如下公式表示。

$$V_{c,\pi}(S_t) = \mathbb{E}[c(S_{t+1}) + \gamma(S_{t+1})c(S_{t+2}) + \gamma(S_{t+1})\gamma(S_{t+2})c(S_{t+3}) + \cdots \mid \pi, S_t]$$

由于 GVFs 可以表达丰富的预测知识，因此被成功的用作辅助任务。

3.3 发现问题辅助任务架构

图 14 中展示的是通用辅助任务架构，以最近一次的观察作为输入，θ为模型参数，输出为任务策略和预先定义的问题的回答 (GVF)。

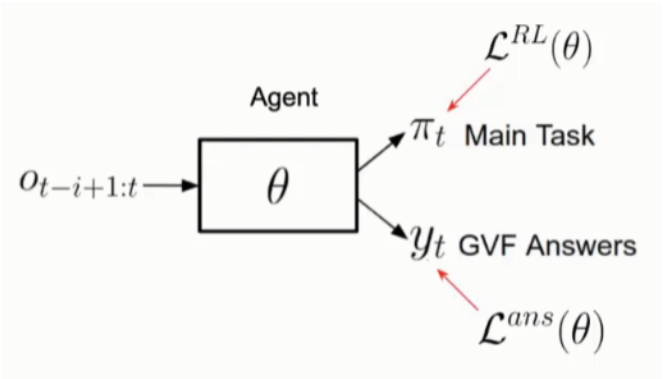


图 14：通用辅助任务架构

在训练过程中，损失函数包含两部分，分别为： 主任务损失函数 $L^{RL}(\theta)$ 和辅助任务损失函数 $L^{ans}(\theta)$。本文中提出将发现问题作为辅助任务，而不只是回答问题，架构如图 15 所示。针对于发现问题，提出单独的问题网络，用未来的观察作为输入，η表示参数，输出为累积向量和折扣因子向量。注意： 未来的观察只能在训练阶段可以获取到，无法在验证阶段获得。但这对于本文提出的方法没有影响，因为本身在验证阶段就不需要问题，只需要在训练阶段提供与回答网络相对应问题的语义表示。

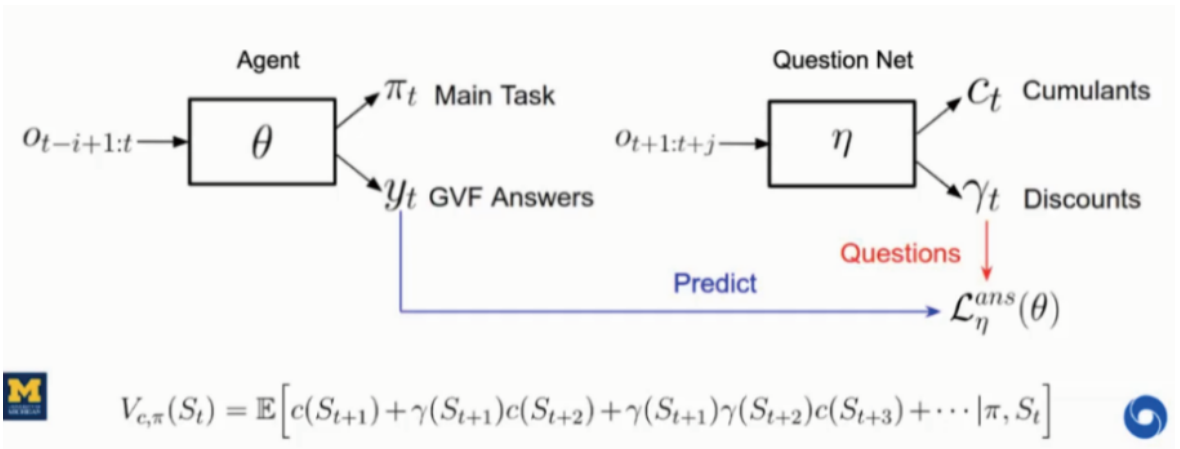


图 15：发现问题辅助任务架构

文中采用 meta-gradients 方法学习更新参数，具体的参数更新过程如图 16、17 所示。

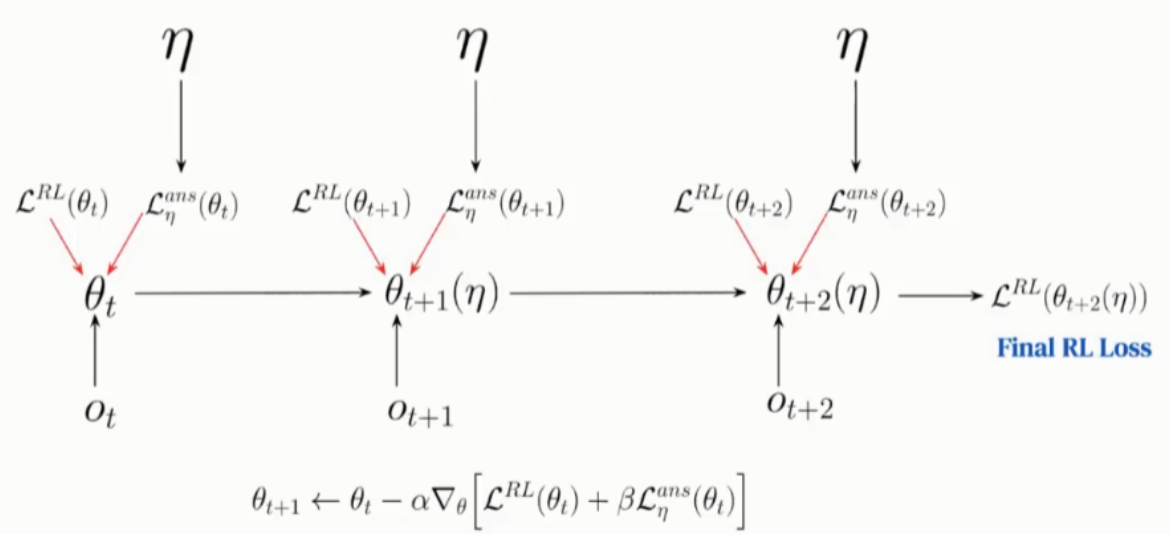


图 16：Meta-Gradients（inner-loop）

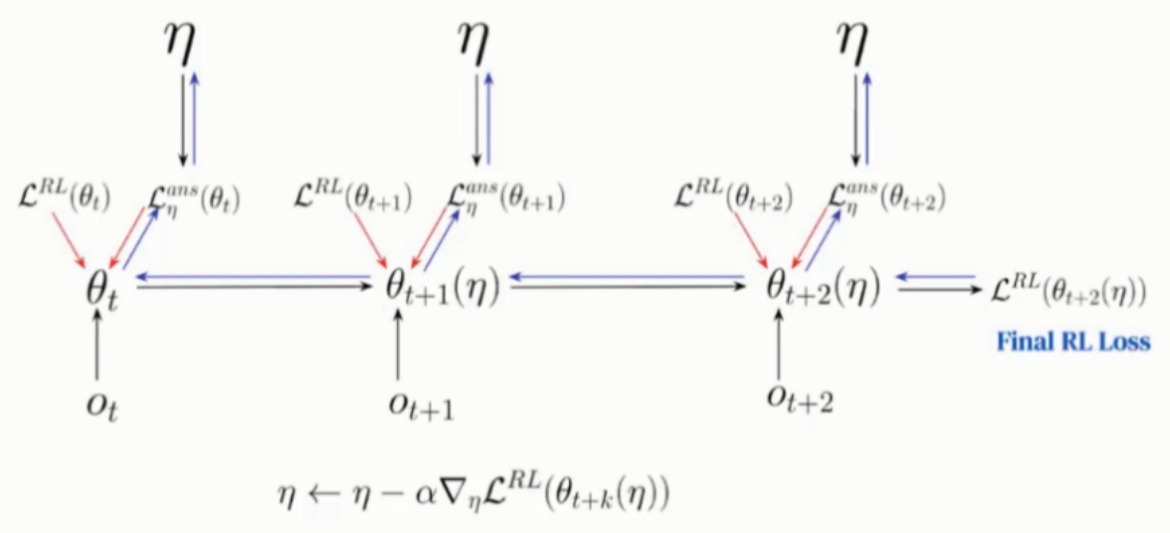


图 17：Meta-Gradients（outer-loop）

3.4 实验

为了验证发现问题辅助方法效果，作者设计了两组实验，分别为： 只有辅助任务学习更新参数（图 18）、主任务和辅助任务共同学习更新参数（图 19），同时设计问题发现辅助任务方法与其他辅助任务方法进行比较（图 20）。实验结果显示采用问题发现辅助任务方法更新参数达到最好的效果。

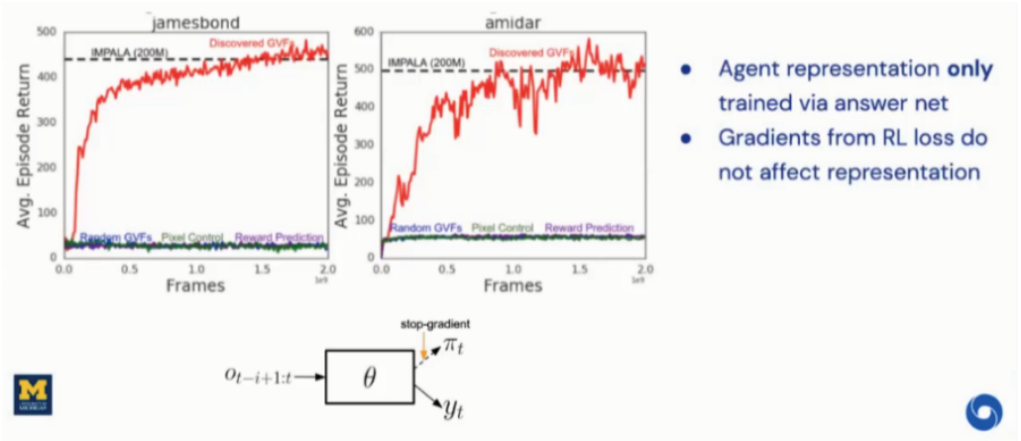


图 18：Representation Learning Experiments

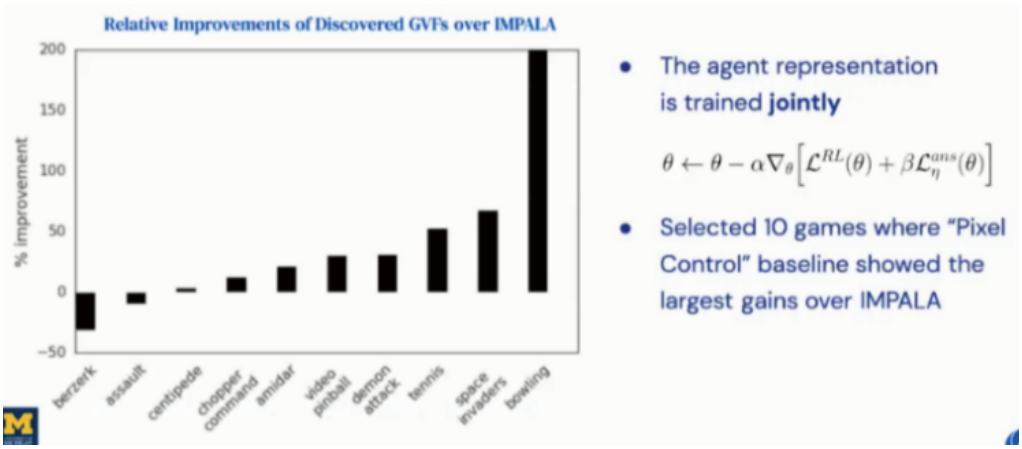


图 19：Joint Learning Experiments

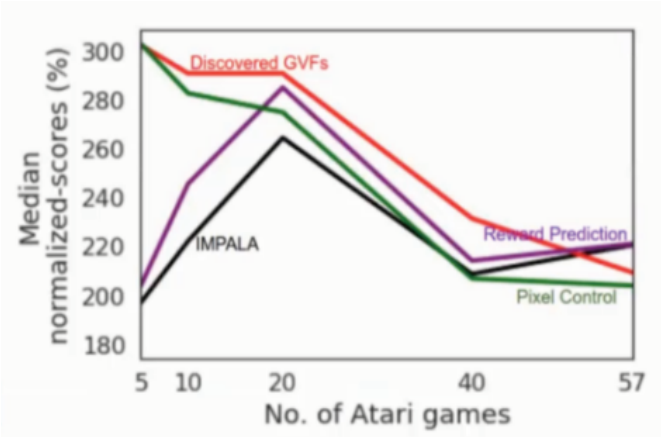


图 20：Comparison of Auxiliary Tasks on ATARI

3.5 总结

文章中提出的方法解决了在强化学习领域如何从自身数据中发现问题作为辅助任务的方法。这个方法可以快速发现问题来提高强化学习效果，但是仍然存在一些限制，比如： 计算长序列问题参数更新受内存大小影响。这也是 Satinder Singh 教授他们未来的研究方向。

四、结语

Satinder Singh 教授分享了他的团队最近的两篇研究工作 [1][2]，讨论了如何将 Meta-Gradient 方法应用的学习发现强化学习智能体中的内在奖励和辅助任务问题中。并通过实验证明通过数据驱动的方式可以发现很多有用的知识来优化强化学习效果。

Q&A

Q1：在强化学习中，内在奖励学习和熵正则化之间的关系？

Singh： 让我用两种方式来回答。第一种是可以通过 Meta-Gradient 方法来学习熵正则化系数。Deepmind 的 Junhyuk Oh 曾经采用过类似的方法，利用反向传播方法学习熵正则化。第二种是熵正则化可以看成是一种质量比较差的探索方法，它无法学习到有用的探索策略。内在奖励学习可以跨多个 epsoides 学习到有用的知识，这一点熵正则化是无法做到的，但它确实是另外一种特别的探索方法。

Q2：你主要关注 Meta-Gradient 框架，请问您有什么理论能保证性能吗？比如：什么场景下效果会更好？

Singh： 简单来说，Meta-Gradient 就是不断的进行梯度计算。我们采用类似 local minimize optimization 等方法保证 Meta-Gradient 性能。但是这些计算是受内存限制的，仍然存在很大的挑战。简短的回答就是： 我们没有很强的理论保障。但是我认为这里有很多有趣的工作值得去做。

参考文献：

[1] Zheng Z, Oh J, Hessel M, et al. What Can Learned Intrinsic Rewards Capture?[J]. arXiv preprint arXiv:1912.05500, 2019.

[2] Veeriah V, Hessel M, Xu Z, et al. Discovery of useful questions as auxiliary tasks[C]//Advances in Neural Information Processing Systems. 2019: 9310-9321.

南洋理工大学安波：竞争环境下的人工智能

整理：智源社区 熊宇轩

当前，以深度学习为代表的人工智能技术已经被应用在了人们日常生活的方方面面。然而，就学术界而言，许多研究仍然停留在通过小规模数据集进行算法验证的层面上，并为涉及到大规模智能系统之间的合作、竞争、通信等问题。在迈向强人工智能的道路中，越来越多的学者试图通过博弈论、强化学习等技术探索竞争环境下的多智能体人工智能系统。在本届智源大会上，来自南洋理工大学的安波副教授带来了题为「竞争环境下的人工智能」的主题演讲，从算法博弈论和强化学习两个方面介绍了他们在该领域的研究成果。

一、竞争环境下的人工智能

在人工智能发展的早期，研究者们主要考虑的是单个智能体 (agent)。到了 80 年代后期，研究者们逐渐开始研究合作的、分布式多智能体系统。到了 90 年代末期，随着互联网、电子商务的发展，人们开始越来越多地考虑竞争环节下的人工智能。当人工智能系统中有多个智能体相互竞争的时候，我们通常会通过博弈论的方法来分析这些问题。

起初，博弈论是数学的一个分支，随后在经济学等领域得到了广泛的应用。今天，对于 Google、微软、百度、阿里等 IT 巨头而言，它们的广告拍卖等技术背后都有博弈论的身影。

到目前为止，已经有了一些将博弈论用于人工智能技术的成功案例。例如，人工智能德州扑克系统几乎完全是基于求解大规模博弈的技术实现的。此外，博弈论也被用于反恐、安全资源的调度、公司的拍卖业务，以及美国的大规模频谱拍卖等方面。

在研究竞争环境下的人工智能时，有两种主流的范式：(1) 基于算法的技术，即算法博弈论。即设计算法来解决竞争环境下的人工智能问题。这类方法有时会涉及近似算法，利用到很多 OR 的技术，尤其是求解大规模优化的技术。(2) 强化学习。这是本次演讲的重点。

接下来，我会分别介绍本团队在以上两个方面做的工作。首先，我们先介绍如何通过算法博弈技术来解决人工智能中有关竞争的问题。

二、冷扑大师

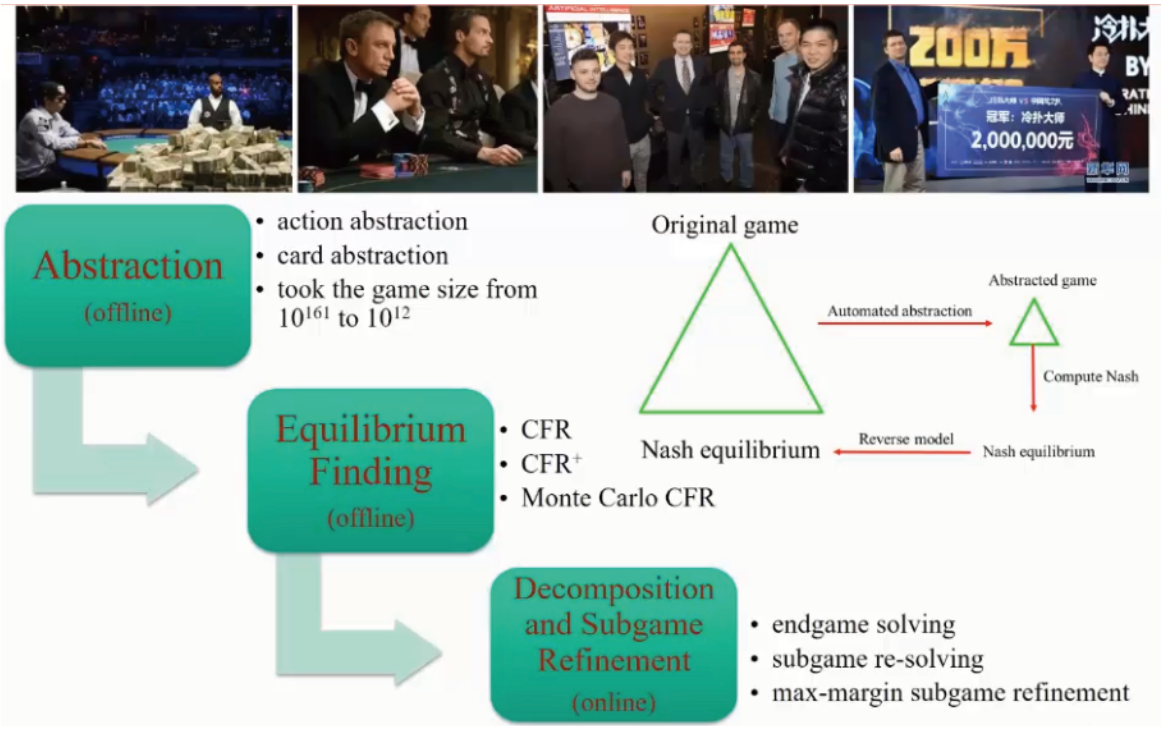
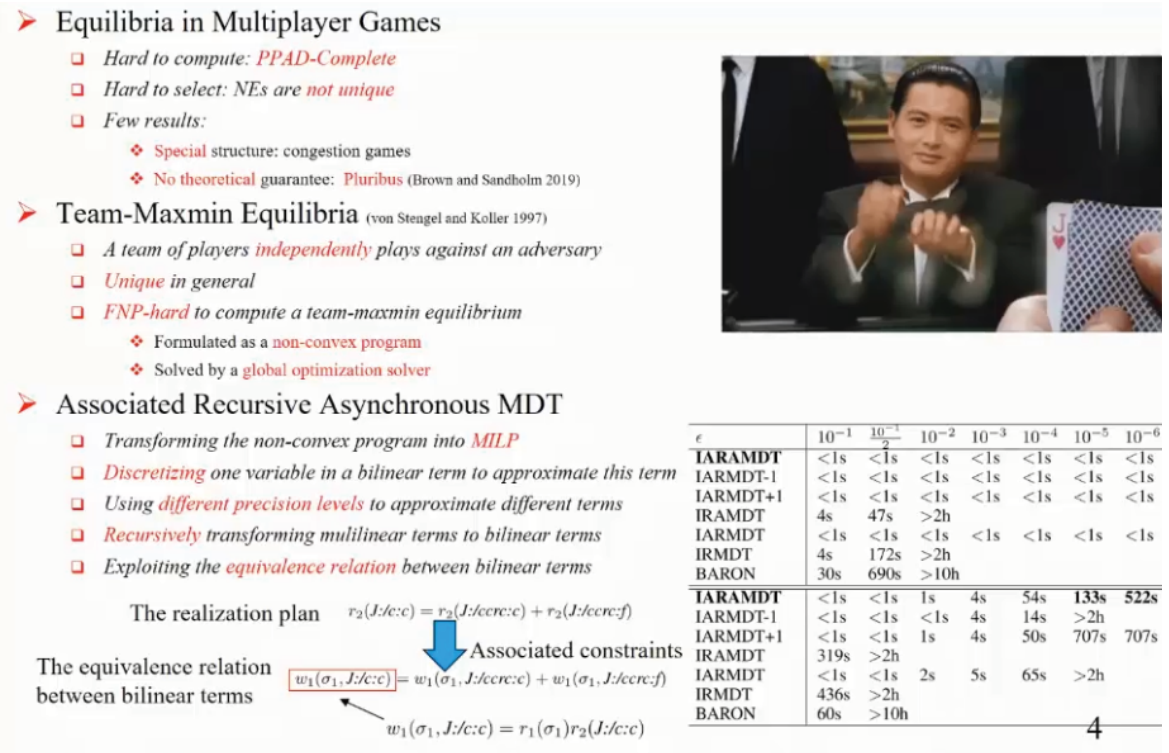


图 1：冷扑大师原理示意图

「冷扑大师」计算机扑克系统可以说是 AlphaGo 之后最为人所知的人工智能成功应用范例。CMU 团队设计的「冷扑大师」的架构如图 1 所示。与 AlphaGo 相类似，「冷扑大师」也需要解决超大规模的优化问题，所以 CMU 的团队采取了很多提升效率的算法技术。例如，他们将原始问题从规模很大的博弈抽象成一个规模较小的博弈，再求解这个小规模博弈。在抽象的过程中，他们应用了很多图异构、同构的概念和技术。在求解这个小规模博弈并得到纳什均衡之后，他们通过逆模型将小规模博弈的纳什均衡映射回原始的问题中。

值得一提的是，这项技术与深度学习并没有关系，这个例子很好地说明了深度学习（甚至是深度强化学习）并不能解决所有的问题。实际上，也曾经有很多团队试图通过深度学习解决与扑克相关的问题，但是其最终得到的效果与使用基于优化的技术所得到的效果仍然有一定差距。实际上，深度强化学习技术至今没有很好地得到理论上的证明。面对一个复杂的问题，如果我们能利用优化技术和基于算法的技术对其进行求解，达到均衡或最优（或者近似最优）的状态，通常会比使用基于强化学习的技术效果更好。

接下来，我会介绍一些本团队近期利用优化技术求解竞争环境下的人工智能问题的一些工作。



ϵ	10^{-1}	$\frac{10^{-1}}{2}$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
IARAMDT	<1s	<1s	<1s	<1s	<1s	<1s	<1s
IARMDT-1	<1s	<1s	<1s	<1s	<1s	<1s	<1s
IARMDT+1	<1s	<1s	<1s	<1s	<1s	<1s	<1s
IRAMDT	4s	47s	>2h				
IARMDT	<1s	<1s	<1s	<1s	<1s	<1s	<1s
IRMDT	4s	172s	>2h				
BARON	30s	690s	>10h				
IARAMDT	<1s	<1s	1s	4s	54s	**133s**	**522s**
IARMDT-1	<1s	<1s	<1s	4s	14s	>2h	
IARMDT+1	<1s	<1s	1s	4s	50s	707s	707s
IRAMDT	319s	>2h					
IARMDT	<1s	<1s	2s	5s	65s	>2h	
IRMDT	436s	>2h					
BARON	60s	>10h					

图 2：计算零和多方扩展式博弈中的 Team-Maxmin 均衡

在被 AAAI 2020 录用的论文「Computing Team-Maxmin Equilibria in Zero-Sum Multiplayer Extensive-Form Games」中，我们在多方博弈问题上进行了一些研究。之前我们提到，在人工智能德州扑克系统刚刚被成功研发的时候，它只能处理两方的博弈，因此其商业价值是有限的。实际上，在网络游戏平台，或现实的赌场中，德州扑克比赛需要有很多人参与，因此研究涉及多方博弈的德扑系统是过去七八年中该领域面临的重大挑战。

在涉及多方博弈的德州扑克比赛中，原先基于条件随机场（CRF）的理论和技术都会失效。尽管去年 Facebook 的研究团队做了一些多人德扑环境下机器与人对抗的工作，但是就技术本身而言，并没有任何的进展，他们还是扩展了之前双人德扑的技术，并且进行了一些真实的对抗。

近年来，许多研究者都试图开发人工智能德州扑克系统，Sandbolm 等人他们首先提出了一种解决方案： 首先，我们把多人德扑分成两个队，即形成一个人跟多个人竞争的局面，而这里的「多个人」可能形成一个团队，这样的博弈叫 Team-maxmin 均衡（TME）。我们采用了很多优化技术，在求解 TME 的问题上做了一系列工作，我们提出的方法的性能要优于所有现有的方法。

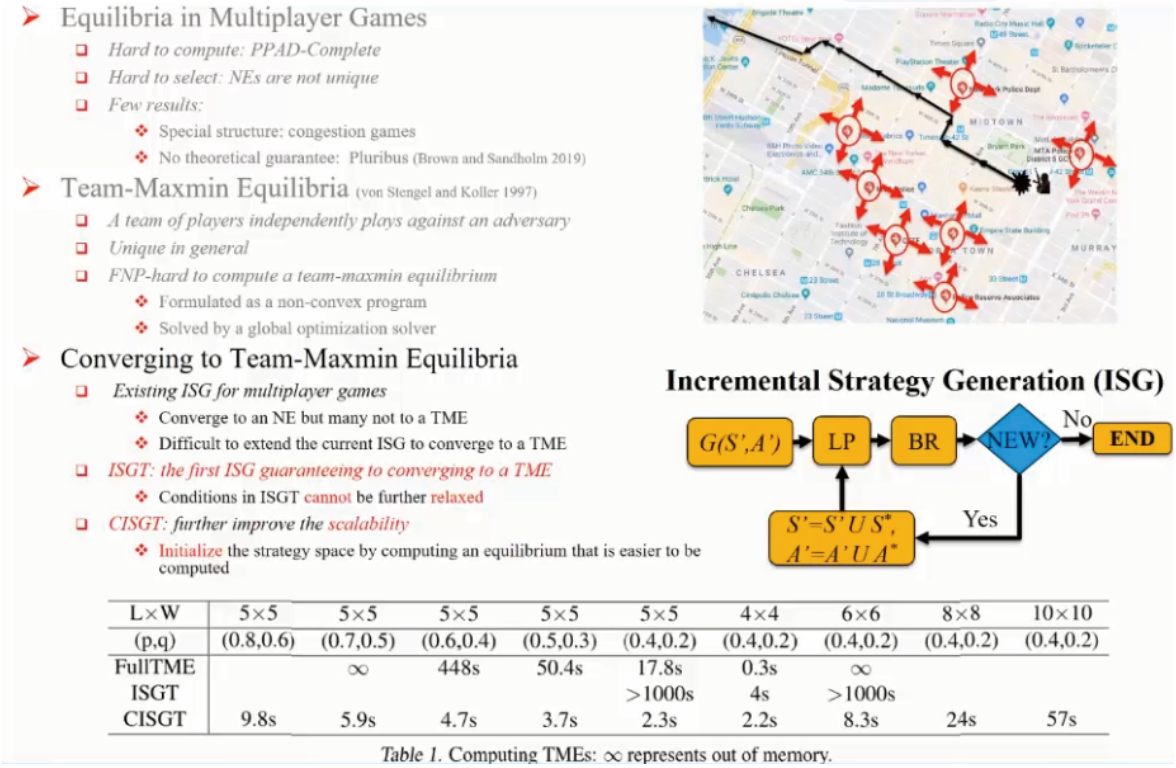


图 3：在多方标准式博弈中收敛到 Team-Maxmin 均衡

在标准式博弈方面，我们于 ICML 2020 上发表了论文「Converging to Team-Maxmin Equilibria in Multiplayer Norm-Form Games」。该论文所面临的主要的挑战仍然是超大规模的动作空间，当博弈涉及大量参与者时，团队的动作空间会随着参与者数量的增加而呈指数式增长。本质上说，我们的目标是找到有效的算法来处理这种超大规模的博弈。

在这篇文章中，我们试图扩展现有的增量式策略生成 (ISG)，但是如果想要收敛到 TME 状态，现有的 ISG 方法往往是无效的。我们采用了 ISGT 和 CISGT 等新的技术保证 ISG 算法能够完全收敛到 TME。

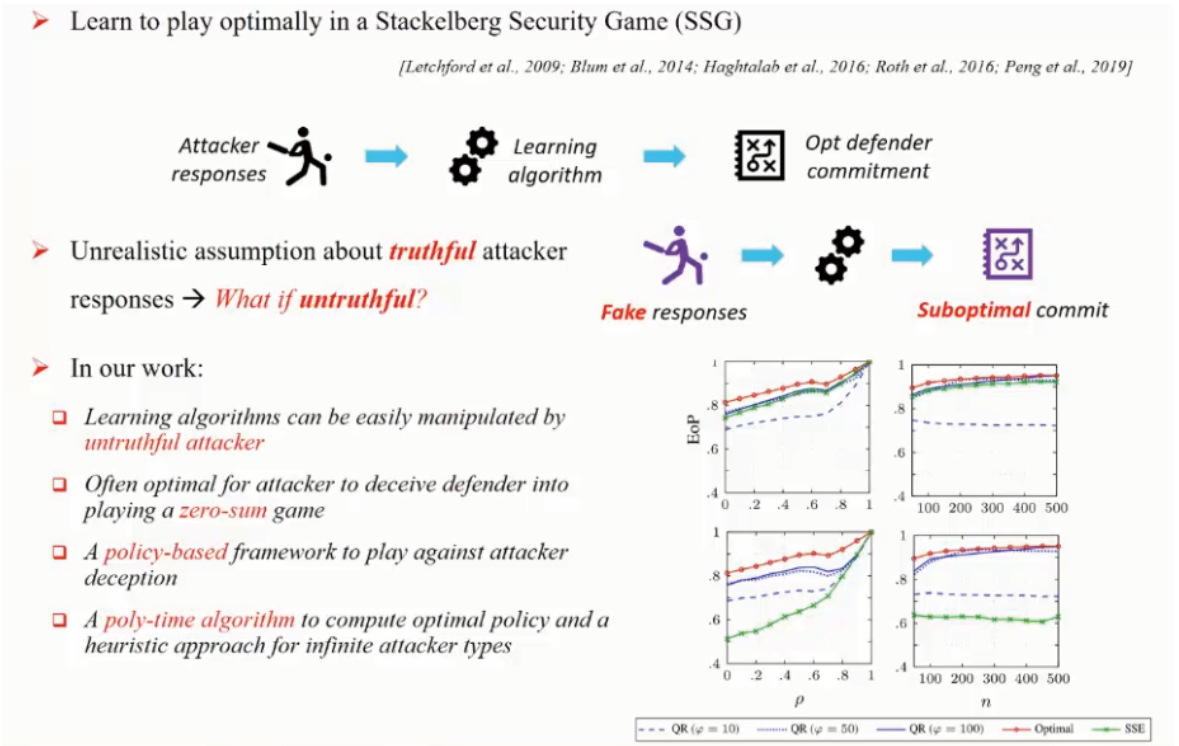


图 4：对抗博弈中的欺诈性行为

在博弈过程中，各方进行对抗时，我们可能会试着去尝试使用一些策略，根据对手的反应和策略获取有关对手的知识。接着，我们进一步利用这些知识来制定策略，达到更好的效果。对于人类而言，我们经常会采取上述策略，有针对性地探测对手的弱点。但是在竞争的环境下，上述策略是较为危险的，对手可能利用我们的学习算法来误导我们，让我们最终得到比较差的结果。因此，我们假设在对手可能误导你的情况下，试图找到一种鲁棒的学习算法。当我们使用这种学习算法时，即使对手知道我们的学习算法，最后我们仍然可以得到较好的策略效果。在我们团队于 NeurIPS 2019 上发表的论文「Manipulating a Learning Defender and Ways to Counteract」中，我们考虑到人工智能系统中的竞争对手可能会采取一些欺骗性的技术，试图能够通过我们的方法得到较好的结果。

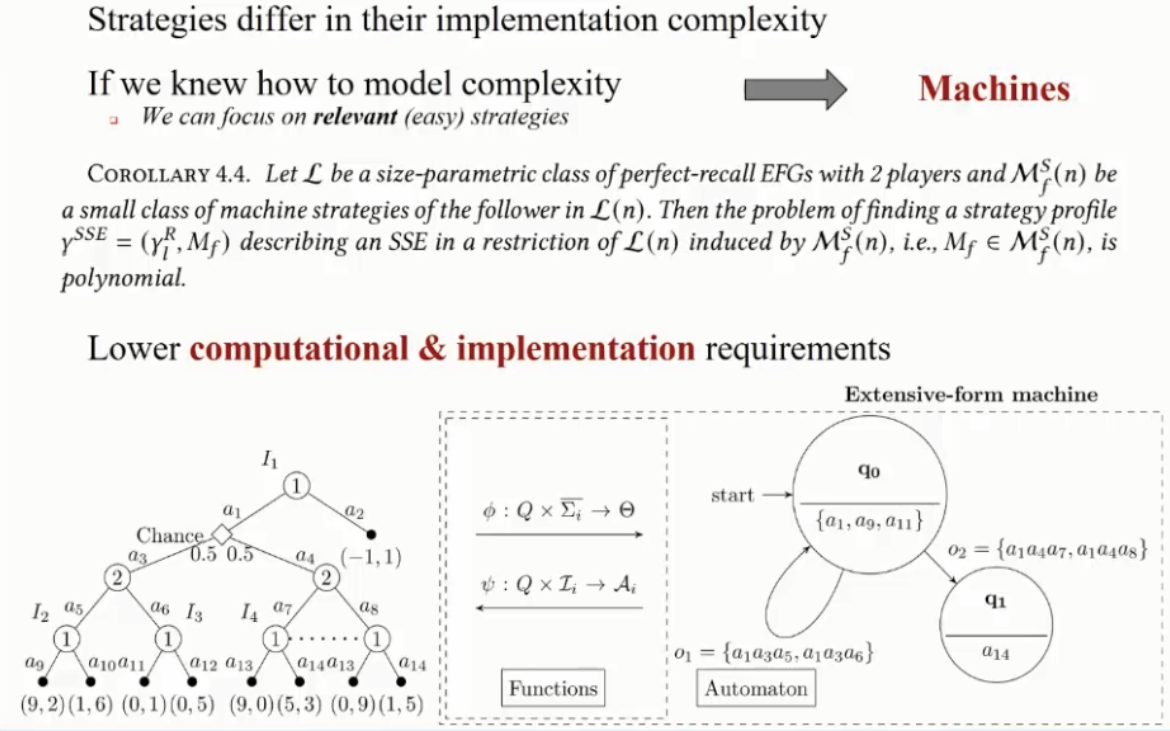


图 5：提升大规模博弈中的可扩展性

为了提升大规模博弈中的可扩展性，我们在 EC 2020 上发表的论文「Playing Games with Machines」中，我们通过自动学习机等技术对每个参与者的策略进行更加紧凑的编码，并且从理论上保证了策略最终得到的结果与真实情况的差距在一定范围内。

三、竞争人工智能环境下的强化学习

在前面的演讲中，我介绍了在有竞争的人工智能问题中，可以采用一些优化技术来解决相应的问题。而对于大多数上述技术而言，几乎最后都存在一个理论的界（bound）来衡量求得的解与真正的最优、均衡之间的距离。针对此类能找到近似的最优解、或近似均衡的人工智能问题，也许使用基于优化或 OR 的技术是最合适的选择。

然而，在很多复杂的有竞争的人工智能场景下，我们可能会用到强化学习（RL）技术。在我看来，RL 可能更适合以下场景：（1）无法很好地对问题建模。对于 AlphaGo 和德州扑克系统而言，尽管存在不确定性，但是这些问题本身是能被完美地建模的。（2）规模。人工智能德州扑克系统最早只能解决两方博弈，如今研究者们尝试求解多方博弈问题。当智能体数量越来越大时，传统的优化技术就越来越难以求解博弈问题。（3）对于非凸或完全不能近似的问题，则难以应用传统的优化技术。（4）无法有效利用领域结构（domain structure）。在大规模博弈场景下，由于计算复杂度的限制，我们无法进行许多任务。然而，我们可以利用很多人工智能问题中的领域结构找到更快速的求解方法。例如，「冷扑大师」应用了很多图同构，异构的领域结构；此外，许多研究者利用了图的性质，试图研究图上的博弈。基于这些领域结构，我们也许能够设计出更高效的算法。

在上述情况下，我们可以考虑使用基于强化学习的技术。近年来，与强化学习以及多智能体强化学习相关的研究热度越来越高。在没有协调器（coordinator）的情况下，智能体之间可能是相互竞争的、可能是相互对抗的，

有可能部分智能体相互竞争，部分智能体相互对抗。在这样的环境下，我们需要确定每个智能体采取的策略，通常我们会采用中心化训练、去中心化执行（CTDE）等架构。近年来，在这个欣欣向荣的领域中，人们提出了各种各样的算法（例如，MADDPG、COMA、VDN、QMIX 等）。

接下来，我会介绍本团队最近年来使用强化学习技术，求解复杂竞争环境下的人工智能问题的相关工作。我们的工作涉及包括游戏、安全、电子商务、城市规划等诸多领域。在有些领域中，智能体之间的竞争会特别激烈，而在其它的领域中竞争则没有那么激烈； 在有些领域中，我们获得的数据很多，但是可能在其它的领域中，数据就会少一点，所以我们会针对具体情况设计不同的算法来解决实际问题。

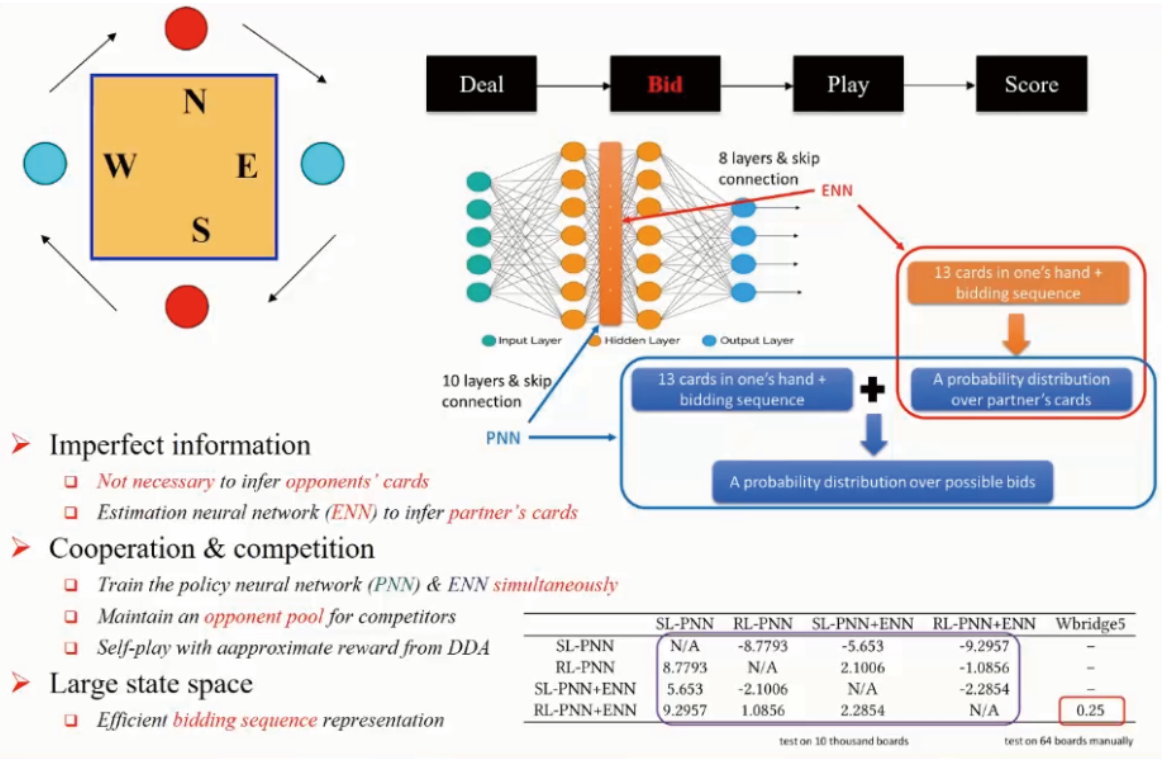


图 6：基于深度神经网络的桥牌系统

在我们于 AAMAS 2019 上发表的论文「Competitive Bridge Bidding with Deep Neural Networks」中，我们以桥牌为场景展开了研究。我们在进行这项研究时，CMU 的队伍还没有尝试研究多人德州扑克。那时，我们认为桥牌比两人德州扑克更加复杂： 桥牌游戏中有四个玩家，这些玩家起码会组成两个队，尽管我和我的合作伙伴会合作，但实际上我并不知道合作伙伴的牌是什么。所以在不确定合作者的牌是什么的情况下，想要更好地进行合作确实是一个挑战。实验结果表明，我们的工作取得了目前最好的效果。实际上，我们采取的方法与 AlphaGo 存在一些相似之处： 我们针对如何出牌设计了一个策略网络（PNN），而我们还设计了另外一个网络来预测合作伙伴的牌，预测结果会成为 PNN 输入的一部分。

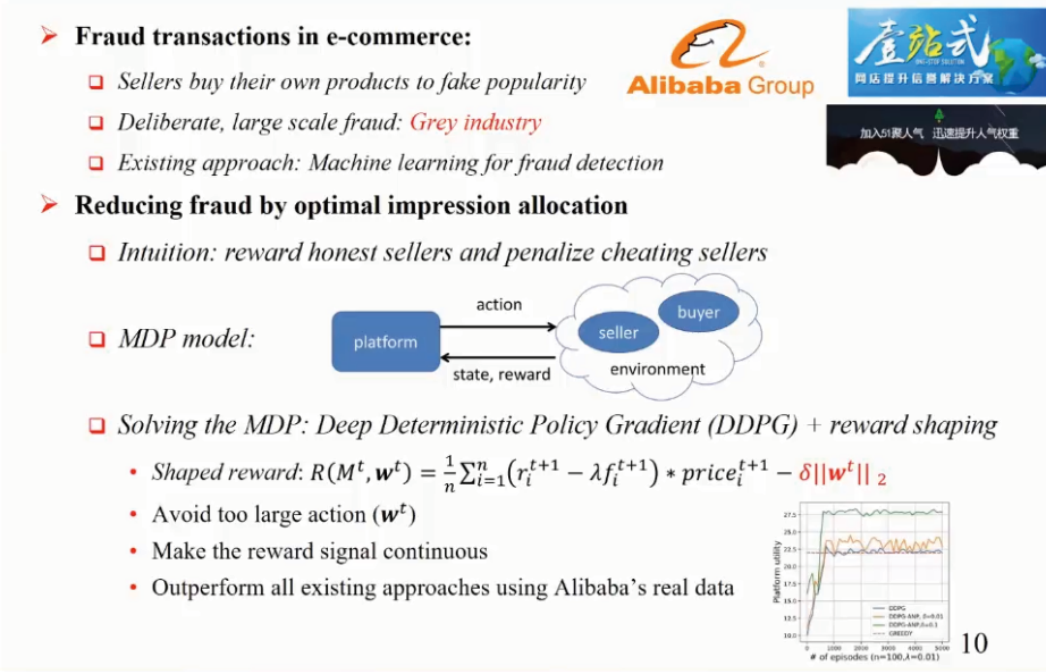


图 7：将深度强化学习用于阻止电商平台上的恶意刷单行为

此外，我们也尝试使用强化学习技术帮助电商平台去制定很好的策略，来优化某些优化目标。在我们于 IJCAI 2018 上发表的论文「Impression Allocation for Combating Fraud in E-Commerce via Deep RL」中，针对淘宝平台上的刷单问题（有很多恶意用户想利用阿里电商平台的推荐系统中的漏洞最大化他们的效益），我们从平台的角度设计惩罚这些恶意用户的策略，从而阻止这些恶意用户参与刷单。为了实现这一目标，我们基于深度确定性策略梯度（DDPG）以及相关技术，学习到了一系列有效的策略，并且成功应用于该电商平台。

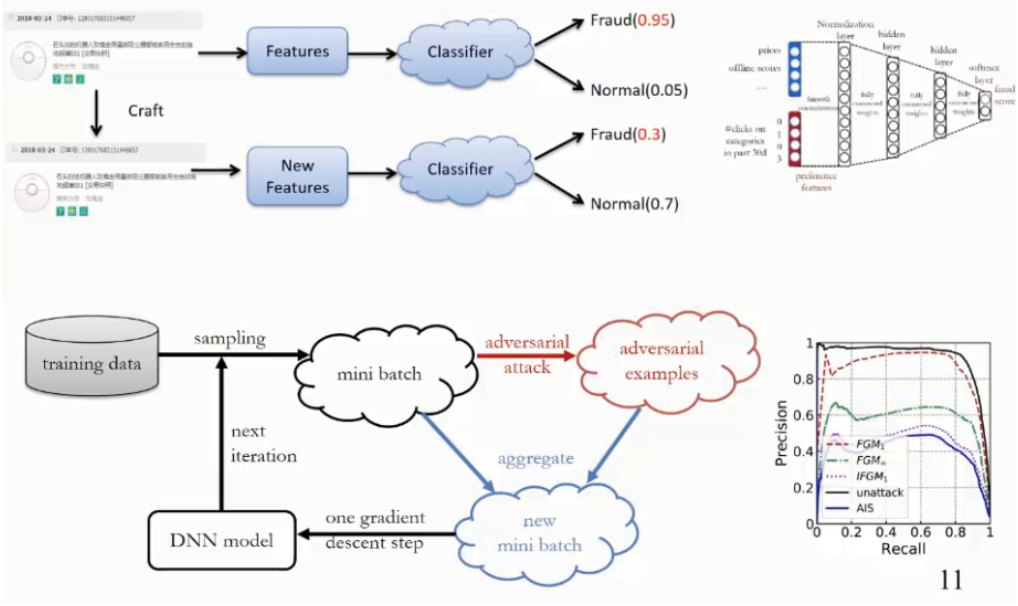


图 8：通过对抗性样本提升刷单检测网络的鲁棒性

此外，在于 WWW 2019 上发表的论文「Improving Robustness of Fraud Detection Using Adversarial Examples」中，我们使用对抗性样本攻击能够侦测交易是否存在刷单行为的网络，根据网络受攻击后的情况，研究如何提升网络的鲁棒性。

尽管这项研究工作于对抗生成网络（GAN）有相似之处，但是当我们要攻击的网络与传统的图形图象识别系统有很大的差别。因为在刷单检测的场景下，几乎所有的变量都是离散的，这导致传统的基于梯度的技术效果都较差。因此，我们需要设计一些新的技术来解决此类问题。

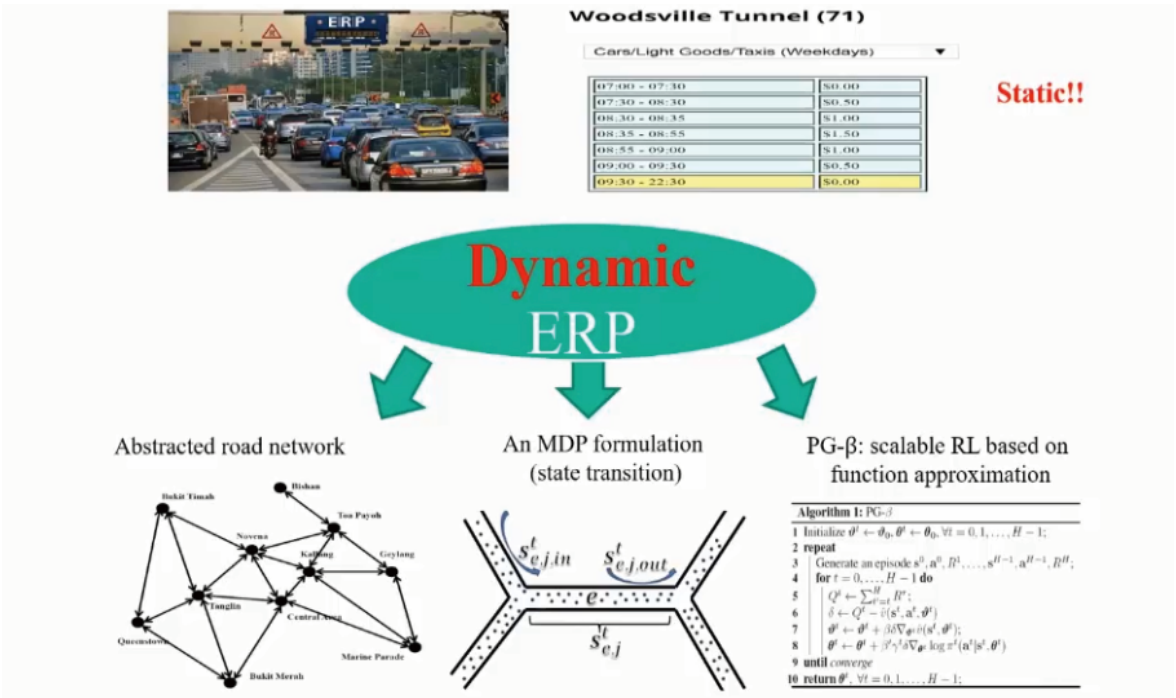


图 9：将强化学习用于城市规划（动态 ERP）

我们也尝试将强化学习技术应用城市规划（例如，电子收费系统 ERP）。具体而言，我们通过 ERP 系统收拥堵费，这种做法在新加坡等地被广泛应用。例如，在早高峰的时段，可能经过市中心的拥挤路段需要花费 30 元人民币。对于传统的基于 ERP 的拥堵收费算法而言，其收费的计算方法是固定的。然而，我们认为可以设计更为合理的收费方式。我们希望能够针对不断变化的交通路况动态调整收费标准：可能某时刻市中心道路十分畅通，但是考虑到 ERP 的收费价格，很多人不愿意进入市中心，是对道路资源的极大浪费。为此，我们提出了「动态 ERP」的概念。研究初期，我们尝试通过强化学习（非深度强化学习）的技术以及一些函数近似方法求解该问题，从而缓解新加坡的城市拥堵现象。在这种复杂的大规模规划问题中，我们需要对所有的需求及其不确定性进行建模。此外，我们还需要对架构变化之后，所有的车选择路径的决策进行建模，这其中也涉及到一些博弈的问题。

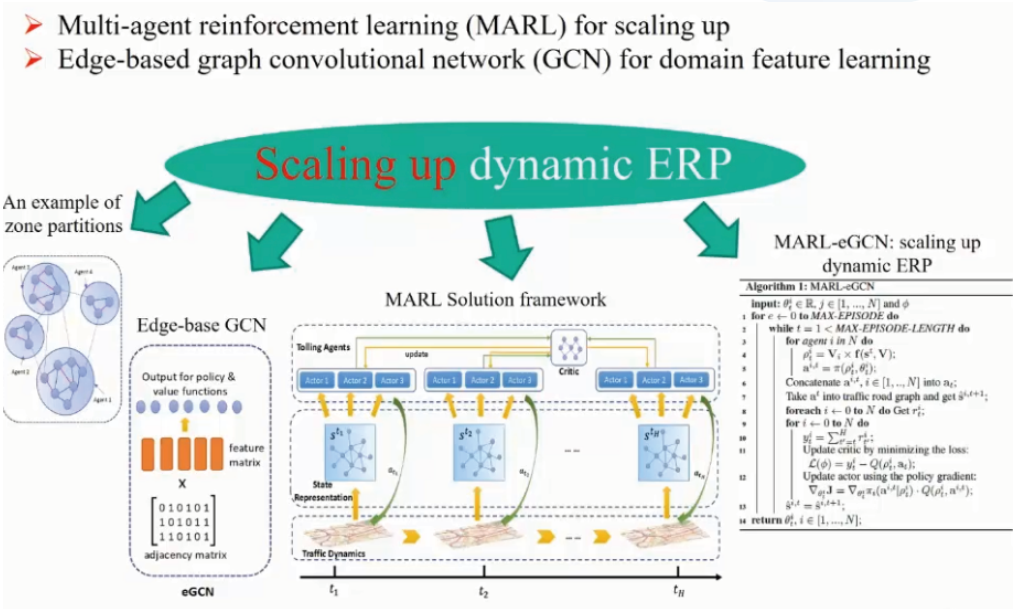


图 10：基于多智能体强化学习的大规模动态 ERP

在 IJCAI 2019 上，我们在论文「Dynamic Electronic Toll Collection via Multi-Agent Deep Reinforcement Learning with Edge-Based Graph Convolutional Networks」中使用多智能体强化学习技术，并考虑了交通问题的相关特质（例如，影响的局部性——某处的交通堵塞开始可能只会影响周边区域，然后慢慢向外传递）。因此，我们试图将 GCN 引入模型，从而更好地对针对该领域的特点进行建模。这些技术放在一起，我们基本上能够解决新加坡的道路拥堵问题。通过动态地对拥堵费进行定价，我们提升了整个交通系统的性能，即减少整个系统的堵塞，或优化其它任何可能的目标。

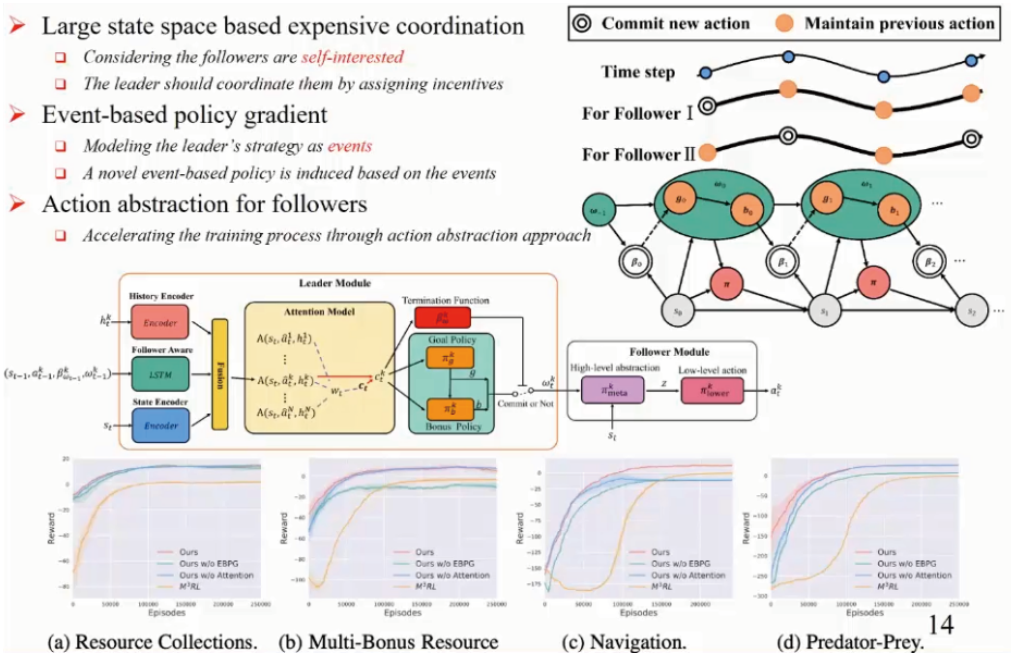


图 11：基于事件的深度强化学习方法

在 ICLR 2020 上，我们发表的论文「Learning Expensive Coordination:An Event-Based Deep RL Approach」考虑帮助政府或公司高层进行决策，这些决策可以激励下层的参与者达到性能良好的均衡状态，最终使优化目标最大化。具体而言，我们的方法会考虑一个制定决策的领导者（政府或公司的高层），这名领导者下面有一些「自私」的、相互竞争的下属。我们的方法旨在在上述情况下为领导者设计一种好的决策方案，令激励下属达到好的均衡状态，从而使整个系统的性能最优。本质上来讲，这是一种机制设计问题，但是我们需要考虑一些更复杂的场景。为此，我们设计了一种新的架构来解决该问题。对于领导者而言，我们将领导的决策建模为一个事件（events），并以此更新基于事件的策略。如图 11 右侧所示，从领导者的角度来说，每隔一段时间会给每一个下属施加一种新的激励措施，或者希望修改其策略。这里的难点在于，我们需要在给定领导者的策略的前提条件下，对下属之间博弈的过程建模。

同样，这里的博弈可能涉及到超大规模的动作空间。因此，我们也研发了一些抽象技术。图 11 展示了该模型的总体的架构，它包含领导者模块、下属模块。在实验中，我们将本文提出的模型跟目前性能最优的一些算法进行了对比。

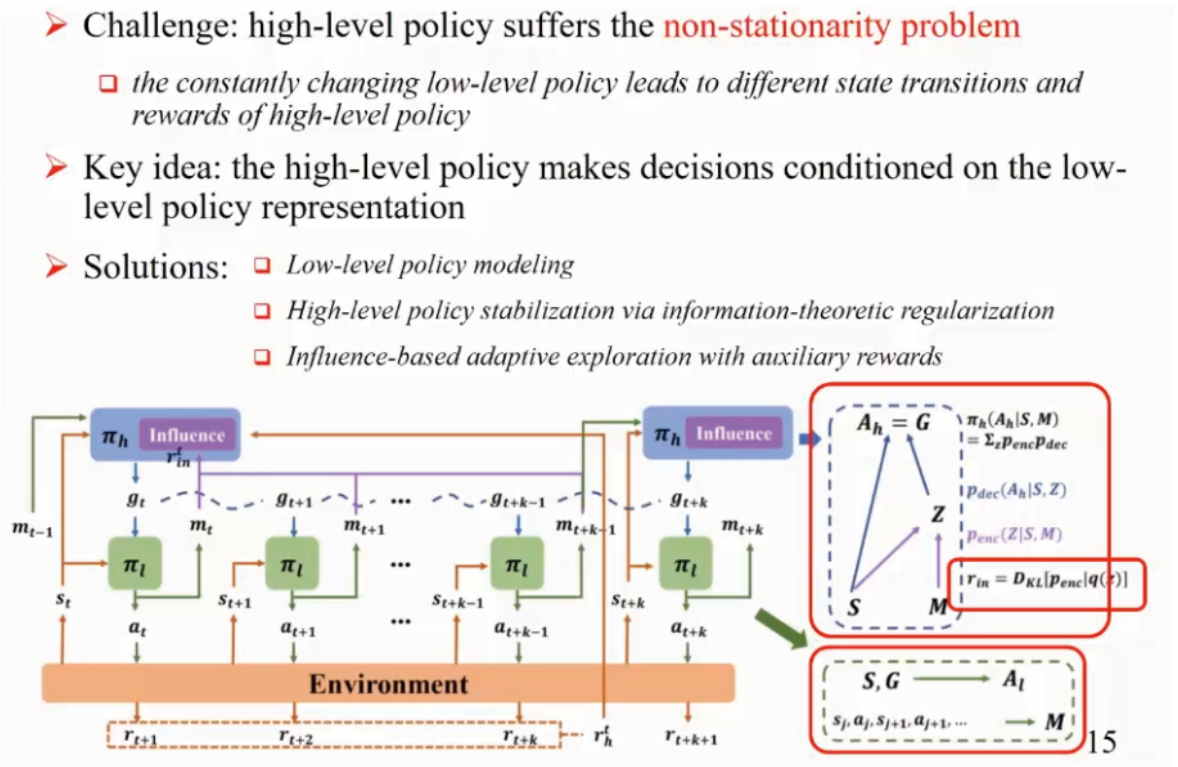


图 12：基于交互式影响的分层强化学习

在 IJCAI 2020 上，我们发表了论文「I²HRL: Interactive Influence-based Hierarchical Reinforcement Learning」，试图解决分层强化学习的场景下，高层的策略通常会存在的非稳定性（non-stationarity）问题。从本质上说，非平稳性指的是低层的策略会不断动态变化。即使高层处于相同的状态（state），在不同的时间步上，低层的策略可能是完全不一样的，这会导致我们得到不同的状态转移结果以及高层策略的奖励。

我们采用了一种简单的方法解决上述问题，即我们希望高层策略会基于低层策略的表征而变化，我们需要对低层策略进行适度的表达，在高层和低层策略之间建立适度的通信，在这种机制之上再对高层策略进行优化。系统的架构如图 12 所示，我们要对低层策略进行建模与表征，然后将这种表征传递给高层策略，从而使高层策略基于这种表征进行更稳定的策略更新。同时，我们设计了相应的技术提升整体策略的探索效果，取得了比目前最优的模型更好的效果。

在我们于 ICML 2020 上发表的论文「Learning Efficient Multi-agent Communication: An Information Bottleneck Approach」中，研究了有关多智能体通信中带宽有限的问题。在许多领域的多智能体通信过程中都存在这一问题。例如，在机器人足球比赛中，会限制每个机器人在一定的时间内，取得相应数量的胜场，这是一种强硬的约束。在这种环境下，我们需要学习相应的策略，决定该智能体应该如何进行通信、与谁进行通信，如何确定获胜等信息。

首先，我们根据「无噪声编码」、「服从高斯分布的随机变量熵值最大」这两个通信原理确定了有限带宽与信息熵之间的关系。基于该理论，我们试图将带宽约束转化为对于熵的约束，试图压缩传递的通信信息的熵。实际上，我们通过信息瓶颈技术引入这种约束。「信息瓶颈」是一种几年来非常流行的技术，它被广泛用于分析深度学习系统中的基础理论。本质上说，我们希望在满足熵约束的同时，传递更多有效的信息。因此，我们通过该约束决定保留怎样的信息，以及调度器（scheduler）如何聚合来自不同智能体的信息并将其分发给其它的智能体，最终进行决策。实验证明，我们的方法也取得了比所有目前最优的方法更好的效果。

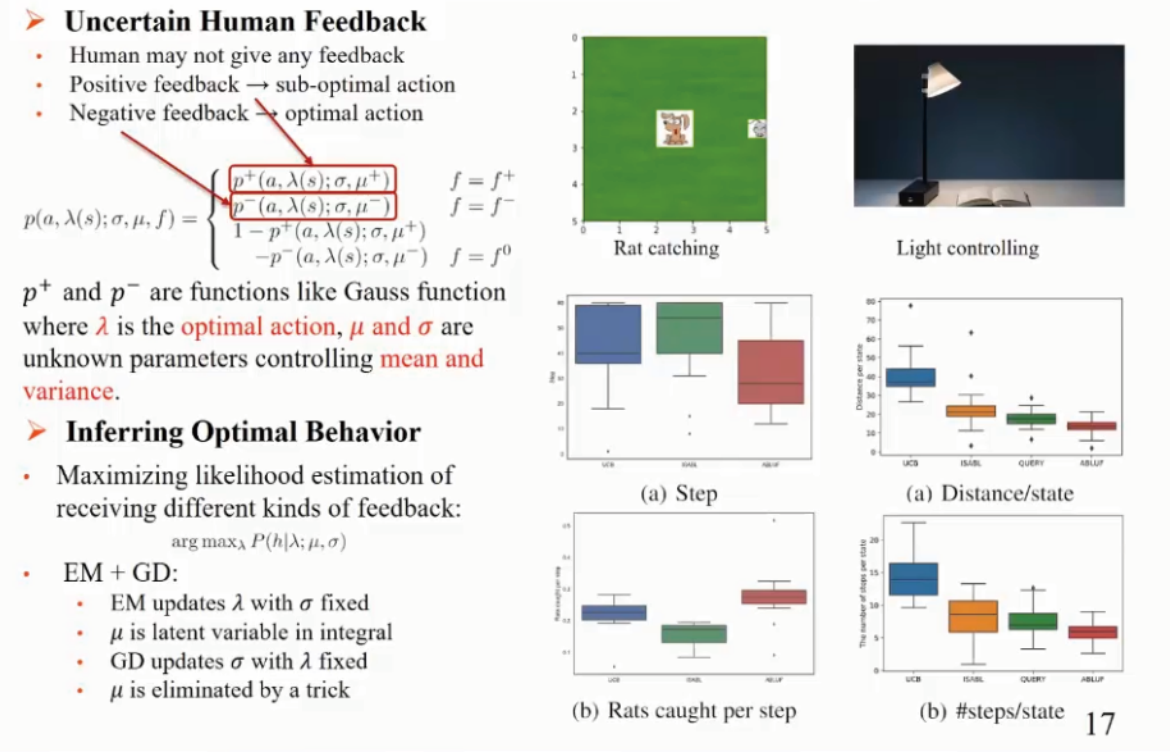


图 13：通过非确定性的人类反馈学习行为

此外，我们在 UAI 20 上发表的论文「Learning Behaviors with Uncertain Human Feedback」中使用强化学习

技术，让智能灯能够根据用户在不同的状态下的反馈，学习如何找出最优的设置，优化其性能，从而使用户满意。在这项研究中，我们仅仅使用了用户的反馈，用户如果不喜欢当前设置，可以对灯做出调整。但是用户有各种各样反馈模型，难点在于确定这种反馈到底如何反映出用户真实的喜好。

图 14：合作式人工智能挑战赛获胜方案

我们的团队之前也参与了一些合作式人工智能挑战赛。实际上，该竞赛也涉及到一些竞争问题。该竞赛比现在的人工智能星际争霸比赛要简单一些，但需要注意的是： 我们需要在该竞赛中进行特别高效的学习。因为在现在的人工智能星际争霸比赛中，我们可以使用模拟器进行无数局训练。那当时我们所面对的比赛场景下，模拟器的速度非常慢，并且不能加速。因此，在状态空间非常大的情况下，我们需要通过极其有限的时间步的训练，找到很好的策略。

四、结语

在我看来，未来的人工智能系统可能需要更多地考虑复杂的环境下的决策问题。例如，环境可能是随机的，可能有多个参与者，可能需要考虑序列化的决策，参与者之间可能存在竞争或合作的关系，还有可能涉及到分布式的优化。

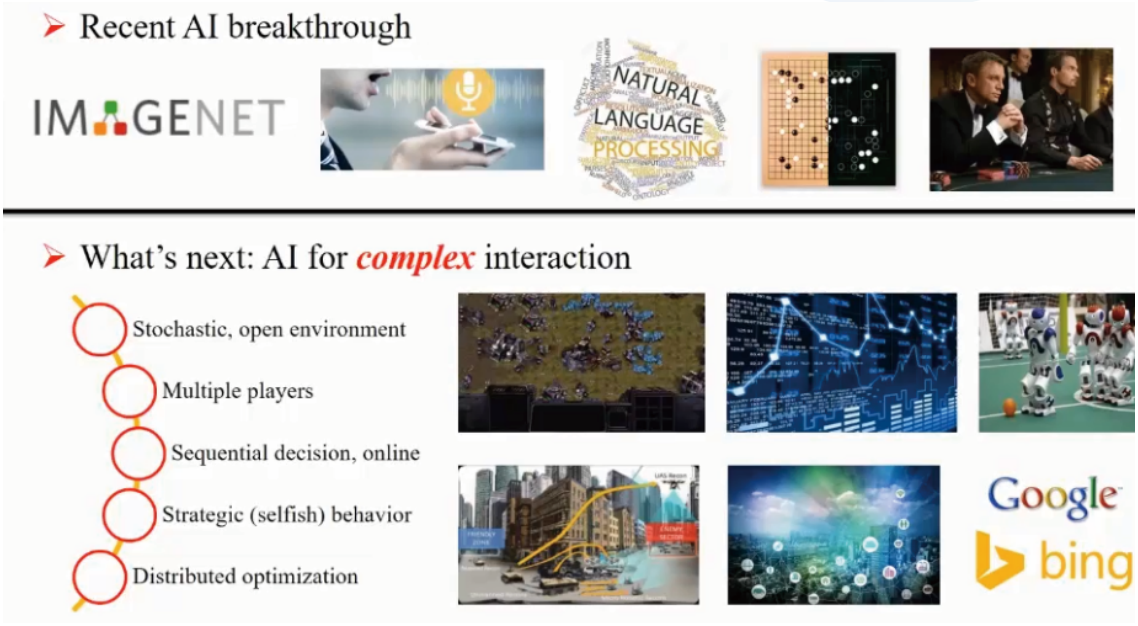


图 15：复杂环境下的人工智能

我认为，在未来，对于解决上述问题来说，多智能体强化学习可能会非常的重要。但是目前来看，这些技术正处于起步阶段，还有很多问题有待解决。目前，多智能体强化学习技术在解决上述问题的时候，几乎仍然完全没有办法把基于优化技术的相关结论和方法集成到解决方案中，该领域的研究仍然面临着很大的挑战。

滴滴 AI Labs 首席研究员秦志伟：深度强化学习赋予网约车交易市场新机遇

整理：智源社区 任黎明

在第二届北京智源大会“强化学习”专题论坛中，滴滴 AI Labs 首席研究员秦志伟博士做了《深度强化学习在网约车交易市场中的应用》的主题演讲。

秦志伟，滴滴 AI Labs 首席研究员，滴滴 AI Labs 强化学习负责人，主要研究方向为强化学习、运筹学优化，智能交通线上运营及网约车交易市场策略优化研究。与团队获得 2019 年度瓦格纳运筹学杰出实践奖（Daniel H. Wagner Prize）。

在本次报告中，秦志伟介绍了滴滴网约车如何利用深度强化学习来全局优化提升用户出行体验和司机收入。

一、滴滴网约车兴起的背景及订单分配的重要性

网约车已经风靡全球，在中国和美国两个市场获得了前所未有的发展。其兴起的主要原因是，随着美国的城市人口已经接近总人口的 82%，中国的城市人口接近总人口 58%，传统的交通方式已经无法满足人们的出行需求，唯一解决的方案就是双边交易市场（Two-side Marketplace）。

在双边交易市场中，司机和乘客都是市场参与者，其中司机关注的是如何增加收入以及如何尽可能地降低空驶时间；而乘客则更关注乘车体验，如司机接驾时间、接驾距离、订单响应率、完成率等。于是作为双边交易市场平台，就需要通过订单分配、司机调度及价格设定来同时优化这些指标。

考虑订单分配的过程。当乘客把出发地和目的地输入到平台后，平台会为每个适用的路线提供一个预计报价，然后生成订单。随后，系统会搜索附近空闲车辆，进行订单分配，如果分配成功，该订单会被指派给一个司机，乘客会被告之驾驶员距离当前位置有多远。如果由于附近车辆需求量低而没有车辆，导致订单搜索和分配耗时太久，乘客可能会放弃该订单；如果司机需要很长时间来接驾，乘客也可以取消订单，寻找其他出行方式。

订单分配事实上并不仅仅是影响到当下司机和乘客的接乘结果 / 短期收益，更重要的还会影响到未来的结果 / 长期的收益。

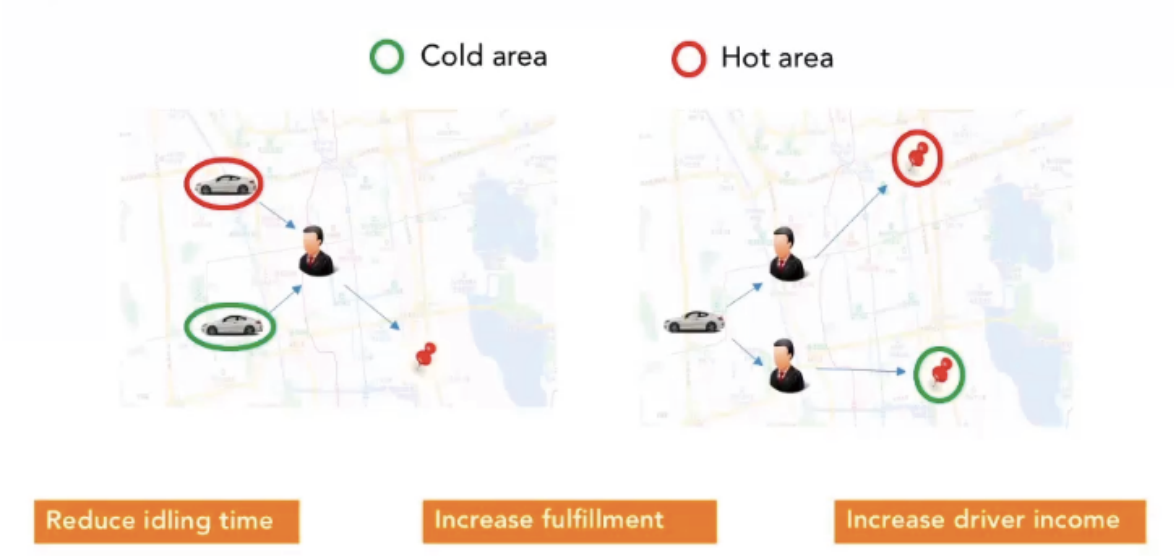


图 1：订单匹配重要性引例

举例来说，假设某地区有冷区（乘客较少）和热区（乘客较多）两个区域，如图 1 左图所示，在冷区，有一位乘客和两辆空闲的车辆（热、冷区各一辆），如果把订单与冷区车辆分配成功，则会降低总体司机空驶时间，因为热区的车辆可以马上接到另一个订单。

再来看右图，有一辆空乘车和两个订单（热、冷区各一单）的情况。两个订单举例都比较远，在司机去接乘的过程中，乘客很容易因为等待时间较长而取消订单。但相对来说，前往热区的订单可以降低整体空驶时间，并提升整体司机收入，因为即使乘客取消了订单，司机在热区也更容易接到新单。

二、泛化策略迭代框架

网约车订单分配问题实际上是一个动态优化问题。首先从需求上来看，用户在任意时间都有可能打车，订单请求是全天候不断进入系统。其次，司机也可以随意上、下线，出租车可用状态是随机的。

解决的办法是在线订单分配 / 匹配策略（π），让订单进入系统之后立刻可以做出分配决策。

$$\pi(o):o\to x\in X\left(t_m(o)\right),X(t)=\ set\ \text{of free drivers at time t}$$

针对这一策略，优化的目标是交易市场的效率，具体则包括，

1）提升整体的司机收入 $max_{\pi}J(\pi)=\sum_{i=1}^{N}p^{i}(\pi)$；

2）订单的完成率 $\sum_{i=1}^{N}1(p^{i}(\pi)>0)/N$；

3) 响应率 N^{+}/N；

4) 减小司机接驾距离 $\left(\sum_{i=1}^{N} d\left(\tilde{l}_o^{(i)}, l_o^{(i)}\right)_{1\left(\tilde{l}_o^{(i)} \neq \varnothing\right)}\right) / \mathrm{N}^{+}$ 等。

这里各个符号的意义可以查看下表:

Symbol	Meaning
l_o	trip origin in coordinates
l_d	trip destination in coordinates
$\hat{p}$	trip price quote
o	order object
t_r	order submission time
t_m	trip assignment (to the driver) time
$\tilde{t}_o$	driver acceptance (of the assignment) time
$\tilde{l}_o$	driver's location at acceptance
$\hat{t}_o$	estimated pick-up time
$\hat{t}_d$	estimated drop-off time
t_o	actual pick-up time
t_d	actual drop-off time
p	actual trip price

图 2：在线订单分配 / 匹配策略符号表

秦志伟提出了基于强化学习的泛化策略迭代框架，来解决上述动态优化问题。

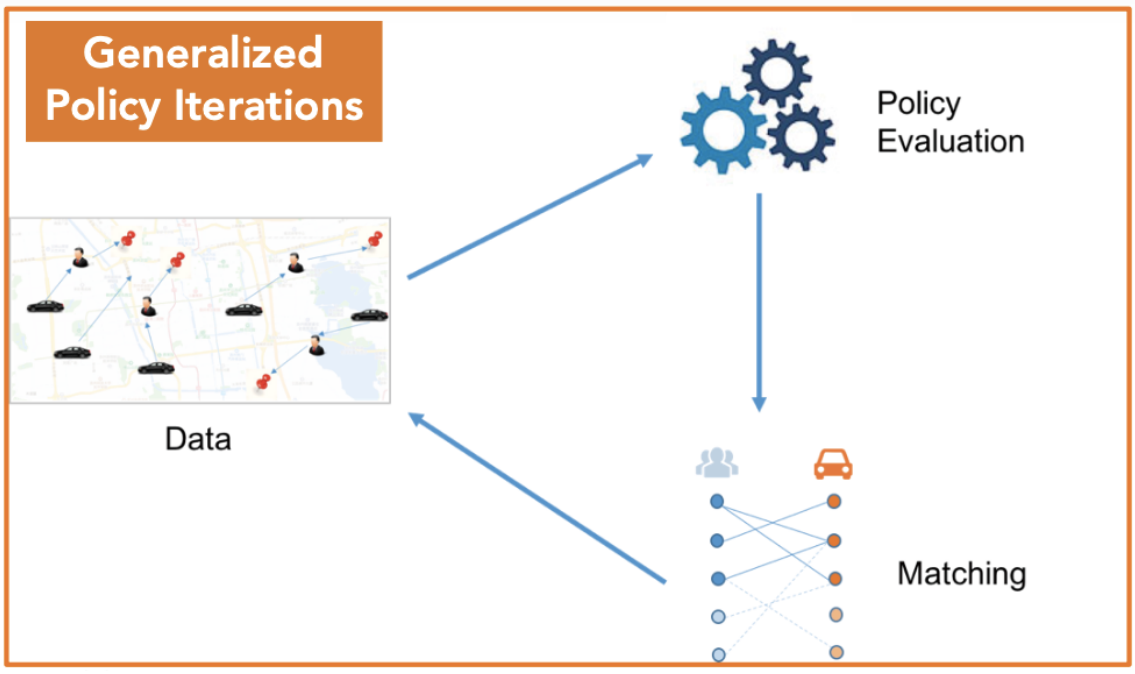


图 3：泛化策略迭代框示意图

首先，在一个既定派单策略下，收集订单交易数据；随后，使用策略评估方法（Policy Evaluation）学习所有数据的价值函数，通过价值网络生成一个新的订单分配策略，然后反复迭代优化。在分配策略输出的过程中，持续迭代深值网络也便于迁移学习，从而能够更好地为多个城市制定出分配策略。

三、行程分配 – 库恩 – 蒙克雷斯（KM）算法

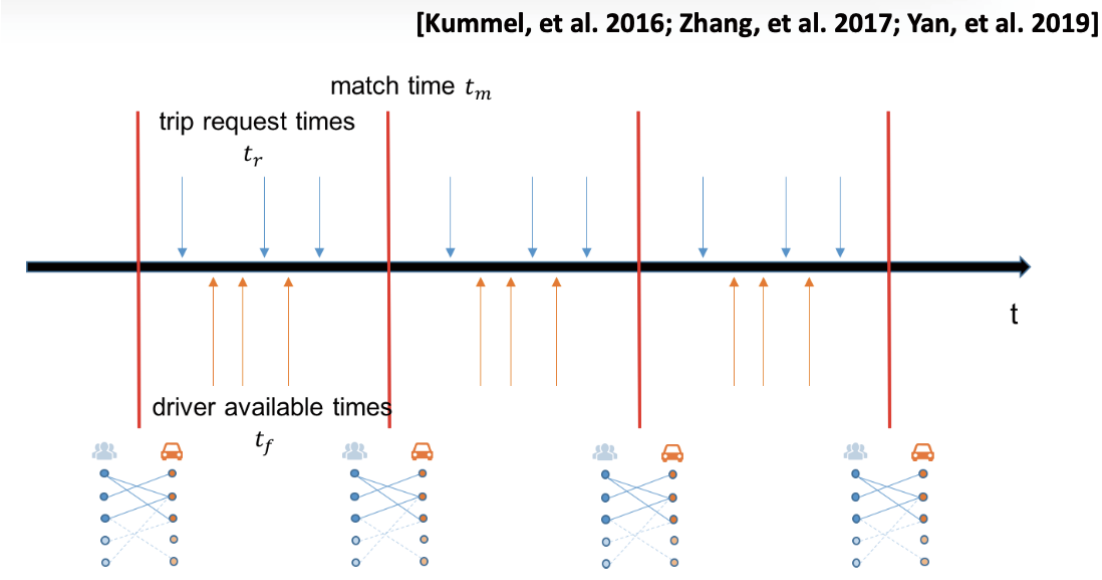


图 4：KM 算法求解动态供需匹配问题

动态供需下的优化订单分配的基本方法是，在预先定义的时间窗口内对订单需求和空闲司机进行批量化处理，并在时间窗口末端（如图 4 右边红色竖线）求解订单分配问题。这是一个线性的分配问题，它可以通过 Kuhn–Munkres 算法或匈牙利算法来求解，其权重（w）可以简单的设为司机和乘客之间的预计距离，从而最小化批量处理内部的总接驾距离。这个数学规划问题可以看做，输入是边权重（w），输出（z）是匹配，寻求恰当的 z 使得策略值最大。

$$\max_{z} \sum_{o \in O} \sum_{x \in X} w_{ox} z_{ox}$$

$$s.t. \sum_{x} z_{ox} \leq 1, \forall o \in O_{disp},$$

$$\sum_{o} z_{ox} \leq 1, \forall x \in X_{disp},$$

$$z_{ox} \in \{0,1\}, \forall o \in O_{disp}, x \in X_{disp}.$$

π

图 5：Kuhn–Munkres (KM) 算法

上述方法得到的策略（π）是短时优化（几秒内）订单分配问题。而从前文提到的冷热区分配问题中，我们知道，在订单分配中长远利益很重要的。因此，我们可以把司机的状态（state，离散时空网格）及动作（option：服务订单或空闲）建模为半马尔科夫决策过程，如图所示。

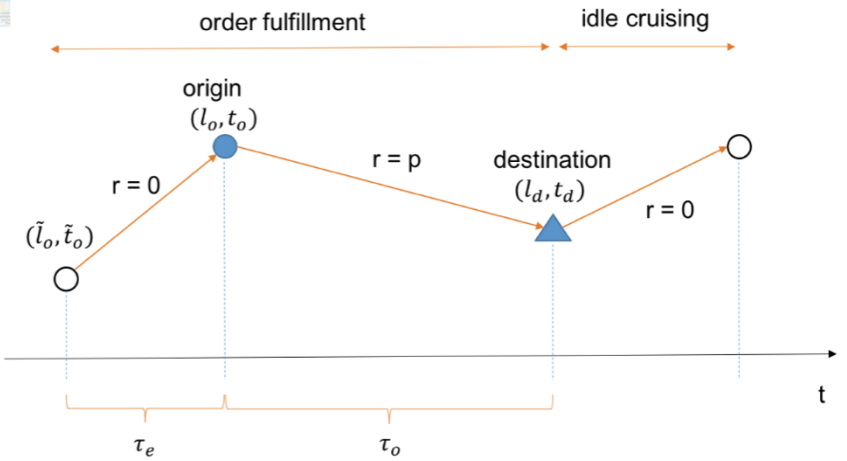


图 6：半马尔科夫决策过程

option 可以跨越多个时间单元。对 option 的即时奖励（reward）是司机行程收入，司机的转移状态是由转移概率控制 $s_{t+1} \sim P(s_{t+1} \mid s_t, o_t)$，在该案例中，转移概率可以认为是固定的。

半马尔科夫决策过程（Semi-Markov Decision Process）是回合制。当状态中时间是一天的最后一刻时，司机的状态为终止状态。如图 6 所示详细描述司机轨迹:

第一段，司机空驶来接驾，即时奖励为 0；
第二段为司机在服务状态中，即时奖励为 p;
第三阶段为新订单的空驶，即时奖励为 0。

Option 的奖励跨越多个时间单元。

把 option 奖励分为段，假设未来时间越远奖励越小，对每段做一个适度的衰减，同时考虑到司机空驶接驾时间（τ_e）所带来的奖励系数γ^{τ_e}），我们可以得到长时间内司机的奖励函数:

$$\hat{r} = \gamma^{\tau_e}\left(\frac{r}{\tau_0} + \gamma\frac{r}{\tau_0} + \cdots + \gamma^{\tau_o - 1}\frac{r}{\tau_0}\right) = \frac{r(\gamma^{\tau_o} - 1)\gamma^{\tau_e}}{\tau_o(\gamma - 1)}$$

司机的策略是从状态映射到订单的一个函数，该策略从个体的角度出发，与系统策略成为对偶关系，即当系统（π）将订单 i 分配给司机 x，司机的策略（π_d）选择接受订单。

$$\pi_d\big(s(x)\big): S \to O$$

$$\pi_d\big(s(x)\big) = o^{(i)} \Leftrightarrow \pi\big(o^{(i)}\big) = x$$

价值函数（Value function）与给定的策略（π）相关联，它是在一个长期累计收入的期望；给定当前司机的状态，它可以从大量历史真实数据中学习出来。

$$V^{\pi_d}(s) := E\left[\sum_{i=1}^{K-k} \gamma^{(t_{k+i} - t_k - \tau_{o_k} - \tau_{e_k})}\hat{R}_{k+i} \,\middle|\, s_{t_k} = s\right]$$

那么如何学习订单策略的价值函数呢？秦志伟提到发表在 KDD 2018 上的一项工作： 表格时间差异学习（Tabular Temporal-difference (TD) Learning）。

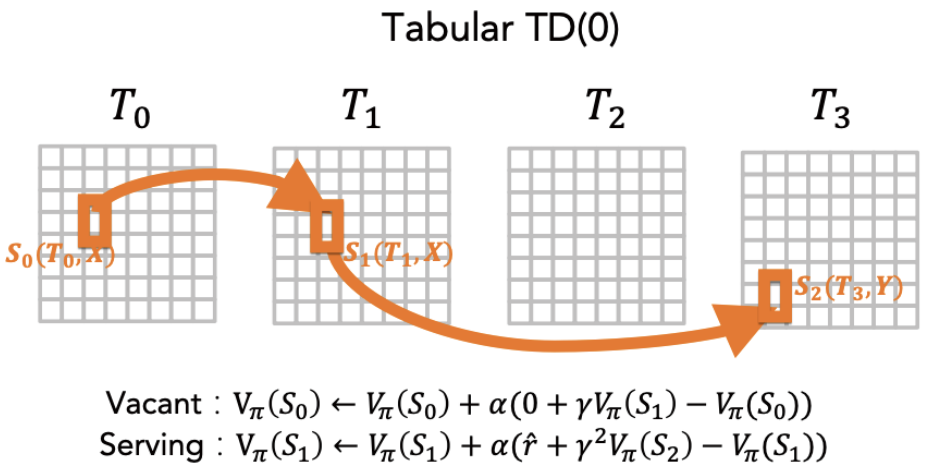


图 7：表格时间差分学习策略评估方法

图 7 中的四个表格（对应图 6 中四个时间点），分别代表不同时间点的时空状态。在 TD 学习方法中每个状态的价值由动态规划更新，状态转移为行程的价值差分，即起始位置价值函数的预测和当前价值函数的差。

于是，我们可以利用当前分配策略的价值函数，通过计算每个司机和乘客的 TD 差，来求得分配问题的边权重。

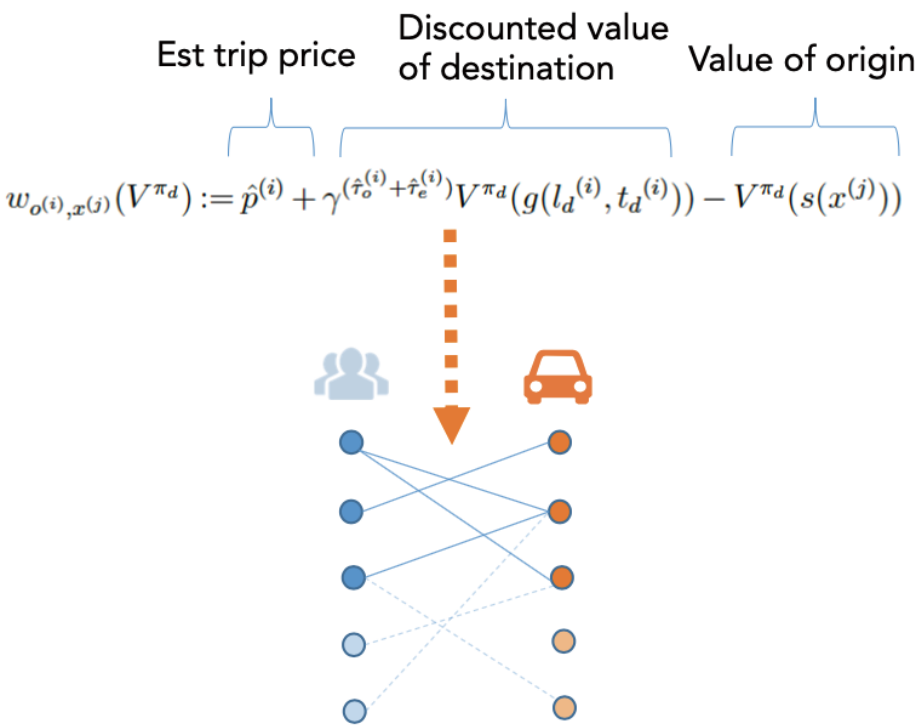


图 8：边权重

边权重ω的定义同时考虑了订单的当前状态价值和目的地的状态价值，有效地解决了前面两个引例中长期收益的问题。

从线性分配问题生成的分配策略并不是对单一的司机贪婪，而是对批量的群体司机贪婪，目标是优化批量司机的整体长期总收入，即司机累计奖励总和:

$$J_x(\pi) = \sum_{k=1}^{K} r_{t_k} = \sum_{i=1}^{N} p^{(i)}(\pi) \Big|_{\pi(o^{(i)})=x}$$

$$J(\pi) = \sum_{x} J_x(\pi).$$

虽然表格式策略评估方法易于使用，但其存在以下三个不足，所以其需要继续完善相关表征和训练方法。

维度诅咒： 表格式表征不能捕获供求动态的特质，需要更多的场景状态来丰富状态表征，表格式价值函数在特征数量上是极少的，其大小会随特征数量的增加而指数增加;
数据稀疏： 训练数据不能很好覆盖整个状态空间，如何将这个价值函数泛化到以前没有出现的状态，得到的结果将会很差;
知识迁移： 平台如何从一个城市的模型训练中提取有用的信息，来支持其他城市的模型训练，从而提高训练的效率和质量？表格式的价值函数缺乏迁移学习的基础，神经网络则比较适合。

四、基于深度强化网络价值学习的订单策略评估

为了克服上述局限性，秦志伟等人提出了基于深度神经网络的价值学习算法 CVNet（如下图所示），来进行策略评估:

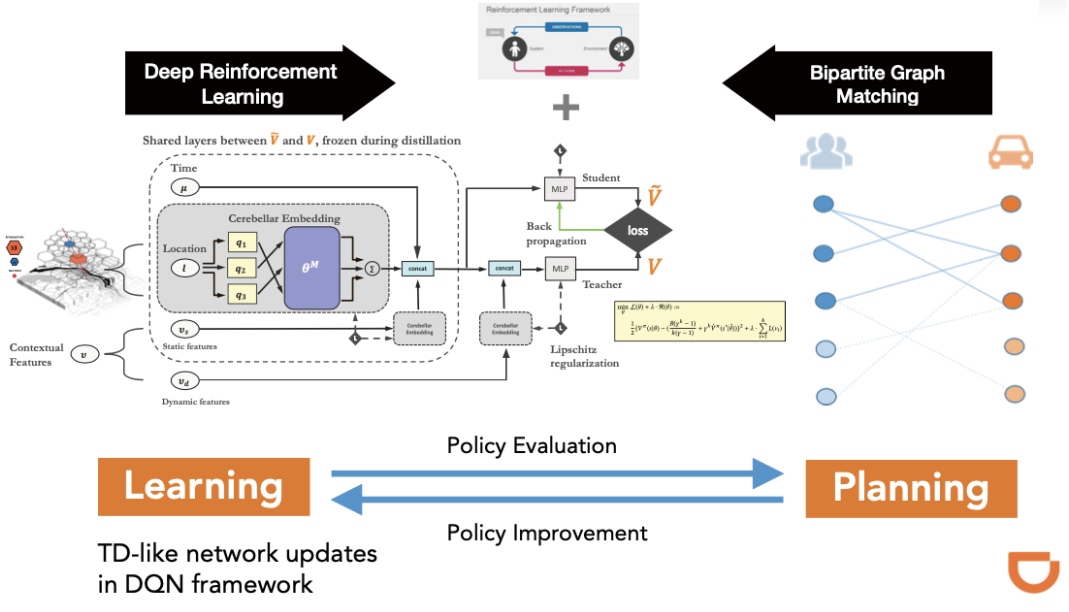


图 9：泛化策略迭代

在状态表征方面，CVNet 能够充分利用层次化的六边形网格系统，来学习自适应空间位置的变化。首先，通过不同层级和大小的格子组合获得特征空间地理位置，每一个位置都能表示为一组不同层级记忆式嵌入的加合，这些嵌入形成了一个高维可学习的向量，它们组成了嵌入矩阵θ^{M}的行，量化单元 q 为激活θ矩阵的行，这些向量行随后会被加和算子聚合起来。

在价值函数的训练方面，CVNet 设计了一个正则化的方法，每个随机梯度下降更新的平方损失项后面加上一个神经网络 Lipschitz 的惩罚项，Lipschitz 常数可以使输入的新函数对输入过程的扰动不太敏感，这对于神经元网络非线性价值函数的逼近很重要，因为在网络更新中每一步都要用到上一步对价值函数的逼近，在随机梯度下降过程中，低质量的更新产生的误差会迅速蔓延至全网络，因此保持一个稳定的值函数对训练的稳定性至关重要。

$$V^{\kappa+1}(s_t) \leftarrow \frac{R_t(\gamma^{k_t}-1)}{k_t(\gamma-1)} + \gamma^{k_t} V^{\kappa}(s_{t+k_t}).$$

$$V^{\kappa+2}(s_t) \leftarrow \frac{R_t(\gamma^{k_t}-1)}{k_t(\gamma-1)} + \gamma^{k_t} V^{\kappa+1}(s_{t+k_t}).$$

Bad value becomes target resulting in more bad values!

图 10：CVNet—Lipschitz 正则化

秦志伟介绍了深度价值网络泛化策略迭代的整体网络架构（如图 9），该网络结构除了未知的输入信息还有时间，动静态的场景特征作为状态特征的输入，场景特征会通过记忆式嵌入与位置特征嵌入拼接在一起，经过 MLP 层输出价值数值。

基于此，我们可以获得一个非常实用的两阶段迭代解决方案，包括基于 DRL 学习阶段和二分图订单分配的规划阶段：

- 学习阶段负责策略评估且价值网络的训练在一个类似 DQN 的框架内进行 TD 类的更新；
- 规划阶段负责基于更新过的价值函数生成一个进阶的分配策略，在分配边权重中体现长期价值司机、乘客的 TD 差，其与前文所述的 TD 学习的情况一致。

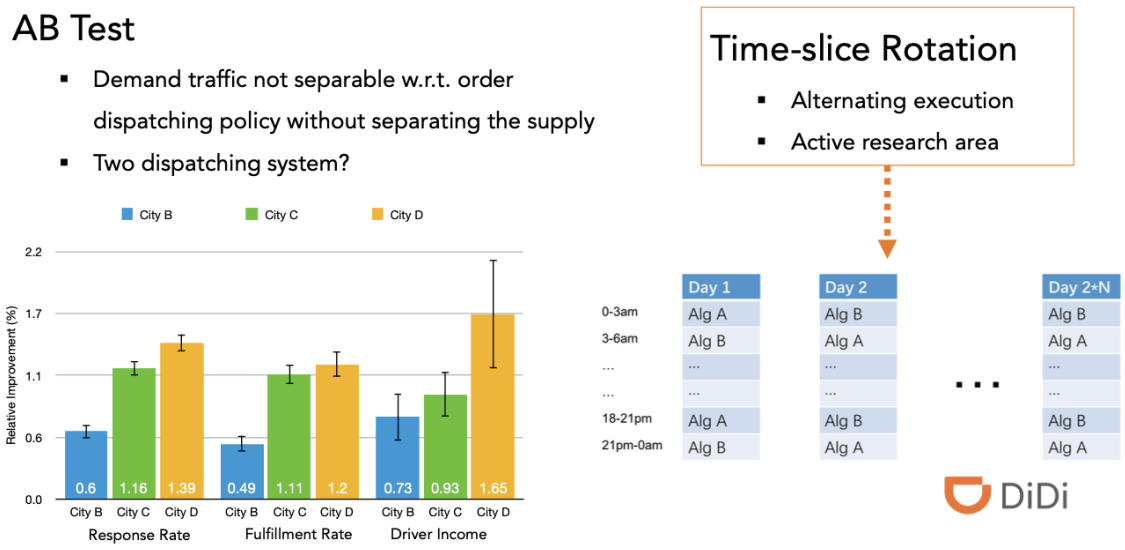


图 11：AB 测试

滴滴公司将此方案部署到三个城市，进行历时一个月的 AB 测试。AB 测试派单算法并不简单，它和测试推荐系统有较大的差异，主要原因该场景为双边市场，虽然订单可以分流，但是司机可以同时接两组的订单，这使得实验无法完全独立分配。若把司机独立会产生两套系统，造成较大的工程成本。所以应该选择其他替代方案，如让两个派单算法在同一个城市运行，并且最小化相互干扰。通过时间片轮转方法让这两个算法在一天中交替执行相同的时间（如 3 小时），且每天执行顺序都在切换，实验持续偶数天。实验结果（如图 11 所示），与距离派单的基线相比，基于 Deep–RL 的方法都具有显著的表现。在司机收入、订单应答及订单成交率指标都有统计学意义的显著表现，这说明司机和乘客的体验都有所改善。

五、重定位策略研究分析及未来研究方向

车辆重定位（复位）策略，即把司机转移到有更好长期收入前景的地方以增加司机整体收入，减少空驶时间。该策略类似于人工智能司机辅助系统，它可以为司机空闲时提供去那里的指示，并且可以告诉司机何时上线或休息。

传统的做法是，司机根据自己的经验来做重定位决策。针对该问题，秦志伟等人建模了两种类型的重定位动作：1）常规重新定位：司机停留在当前的区域或到相邻区域，其涉及相对较短的订单距离；2）长时间搜索重定位动作：订单可能会超越当前区域，把司机带到更远的地方，但其奖励的期望值更高。在订单分配策略中假设所有的状态空间情况相同，重定位动作的奖励是由订单重新定位策略的成本和费用决定。

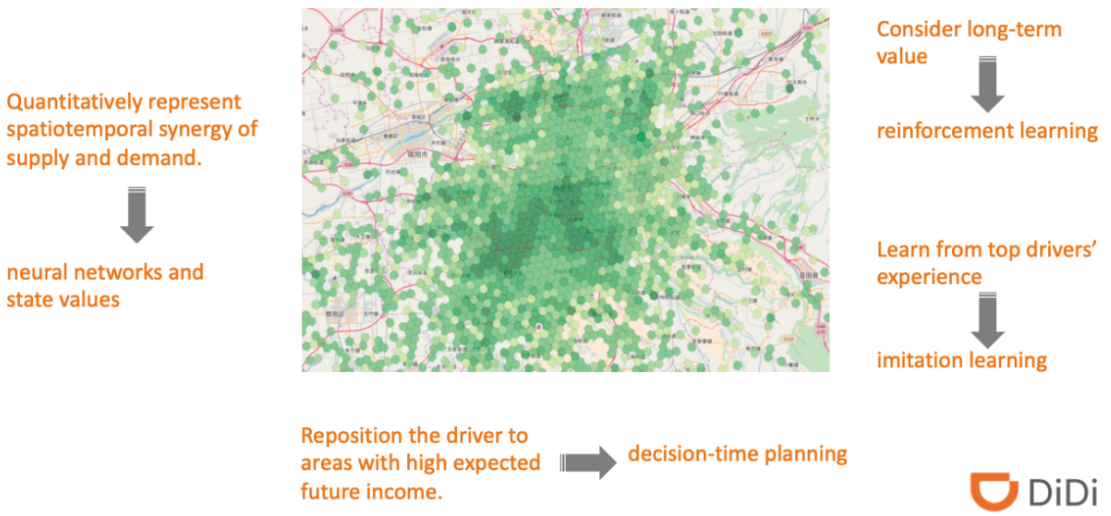


图 11：核心思想

由于重新定位的模型是动态的，因为司机在空驶状态进行重定位动作，且在重定位过程中平台可以随时将订单分配给司机，司机也可以在任何给定的状态下停留，这些可能会导致平台的成本增加。

为此，秦志伟提出决策 – 时间规划（Decision–Time Planning）模型来评估订单分配策略。在决策 – 时间规划模型中，司机在状态 s 订单分配的概率为 $P^{(s)}d$，空驶的概率为 $1–P^{(s)}d$，重定位的当前状态为 S_0，目标状态为 S_i，订单服务预期时间间隔 ETA 为 $\Delta t(s_0, s_i)$。

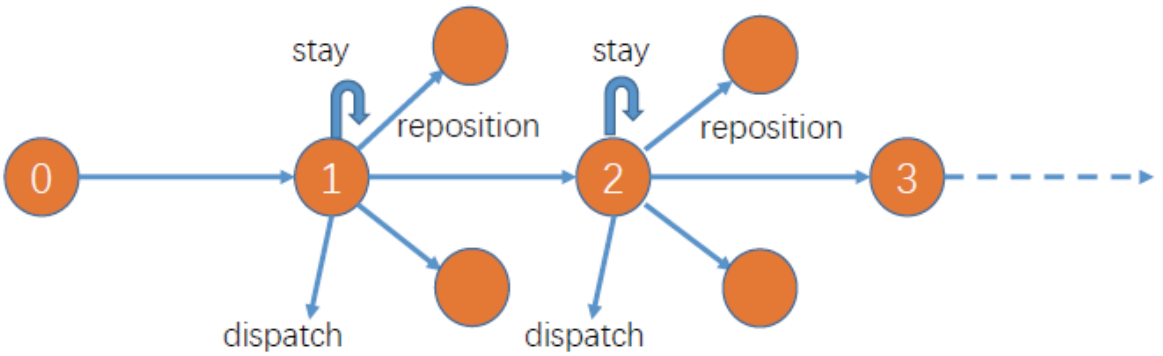


图 12：决策 – 时间规划的环境模型

为了在决策时间计算出最优重新定位策略动作，该模型会基于价值的策略搜索方法，来评估重定位策略动作的状态价值 $\hat{Q}(s,a)$ 的优劣（见下式）。

$$\hat{Q}(s,a) = r + p_d^{(1)} V(s_1|dispatch) + p_{id}^{(1)} V(s_1|idle)$$

Conditional V-Net, $\tilde{V}$

$\max_j \hat{Q}(s_1, a_j)$

Expand by reposition policy

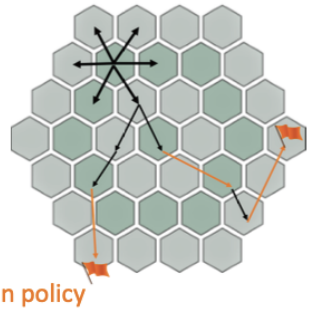


图 13：估计 Q 值

其中 Conditional V-Net 是指保持分配数量相同情况下，使用 V-Net 学习上式中 Conditional V-Net 的状态价值 $\tilde{V}$，$\tilde{V}(s)$ 表示状态动作 $\mathrm{Q(s,o)}$ 的价值函数。通过下式进行双重策略评估，即同时训练 $\mathrm{Q(s,o)}$ 和 V(s)（见下式），并使用与 V 和 Q 网络相同的状态表征层和双重分类器更新优化 V(s) 和 $\mathrm{Q(s,o)}$。

$$\mathrm{V(s)} \leftarrow \mathrm{r} + \gamma^t V(s')$$
$$\mathrm{Q(s,o)} \leftarrow \mathrm{r(o)} + \gamma^t V(s')$$

基于价值的策略搜索方法，通过下列三个式子，对训练过程进行调整可以获得新状态最优动作的状态值，并且每一步都会产生近似最优 Q 值的样本值，以扩展重新定位策略。

- $\hat{Q}(s,a) = r + V^{(t_{01})}(s_1) := r + V_1^{(t_{01})}$, where $r \leq 0$ is reposition cost. -> **Greedy method (one-step policy improvement)**
- Two-step: $\hat{Q}(s,a) = r + p_d^{(1)} \tilde{V}(s_1) + p_{id}^{(1)} \max\left\{\max_{j \neq 1} r^{(j)} + V_j^{(t_{0j})}, V_1^{(t_{01}+1)}\right\}$
- Three-step: $\hat{Q}(s_1, a_j) = r^{(j)} + p_d^{(j)} \tilde{V}_j^{(t_{0j})} + p_{id}^{(j)} \max\left\{\max_{k \neq j} r^{(k)} + V_k^{(t_{0k})}, V_j^{(t_{0j}+1)}\right\}$

此外，向平台上的最优秀的司机学习，也很重要。每位司机关于城市道路拥有极强的领域知识，这些知识可能很难从价值函数中推断出来，例如，有些地区可能更容易停车和接驾。充分利用拥有高收入及高完成率的司机的空驶轨迹数据，从而学习重定位的随机策略。

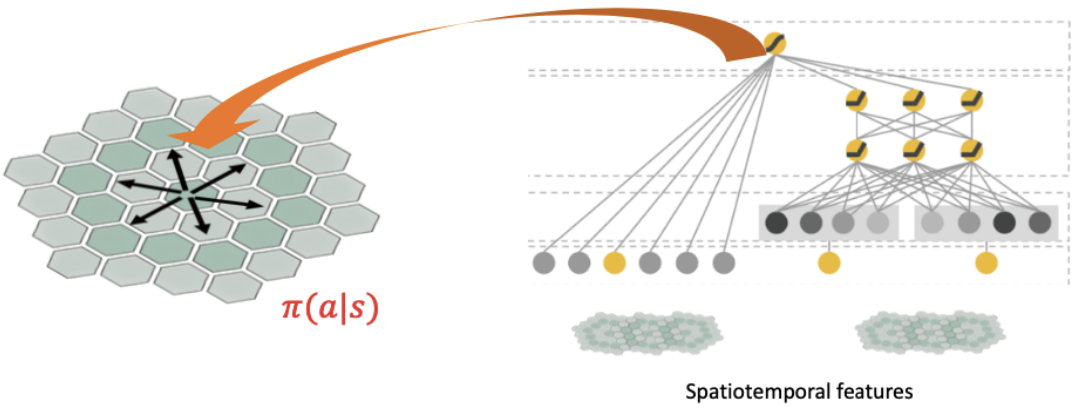


图 14：充分利用司机经验学习时空特征

此外，针对行程较长的订单，会把司机送到郊区或低需求的区域，当应用邻近区域重定位时，效率就会很低。针对这一问题，秦志伟等人提出了单步预测（贪婪算法）的方案。该方法可以搜索更大的区域半径以提供更多的订单策略，形成一个订单状态价值（如订单完成时间最短）排列的目的地候选集。但该算法进行重定位时搜索订单时间会比较久，并可能要求司机到一个存在更高需求的地区（在该区域可能会有连续的订单会分配给司机）。

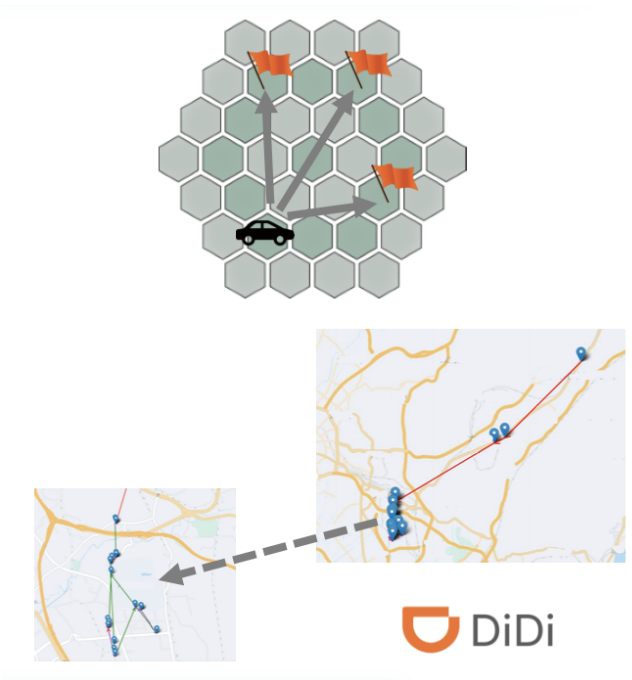


图 15：长程搜索

因此，在重定位策略中，根据司机所处的当前状态及不同的场景会形成三种不同时间决策规划策略：

1. 当订单较多时，模拟历史高绩效的重定位策略会被触发；

2. 当订单目的地相差不大时，高价值搜索订单重定位策略会被触发；
3. 当司机在低需求且没有明确目的地的区域时，长期搜索订单重定位策略将会被触发。

秦志伟等人将上述方法在中国南方的两个主要城市进行了 500 个滴滴司机试点。

算法组：报名的司机能够收到算法重定位的提示；
对照组：从普通司机群体中随机取样（自由空闲驾驶）。

实验连续进行三周，从周一至周五，每天上午 11 点到晚上 7 点。

该试点项目的结果以收入率和利用率为评价指标。其中收入率是单位在线时长的收入，即总收入除以总在线时长；利用率是总在线服务时间与总在线时间的比值。

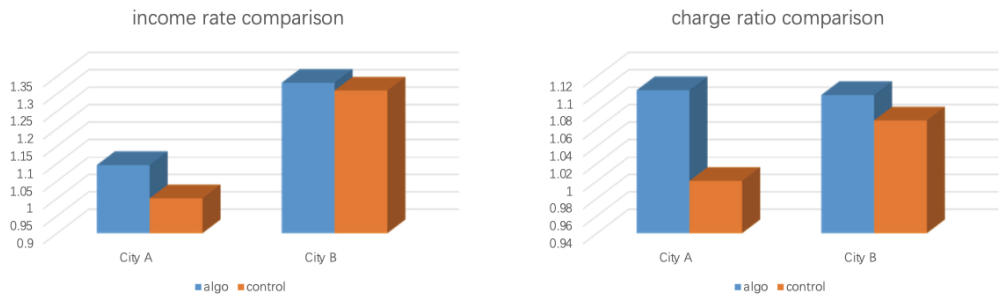


图 16：实验结果分析

如图 16，三个星期的实验，在两个指标上，算法组的结果总体都显著好于对照组。

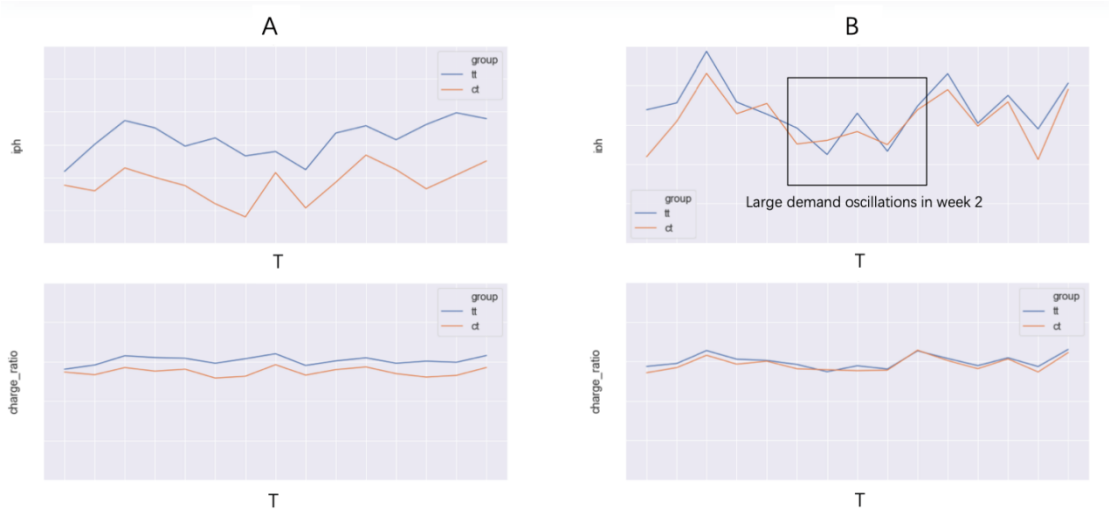


图 17：每天的实验数据对比图

图 17 是收入率和利用率随时间的变化。秦志伟团队发现，在实验中下雨天气 (B 城市) 会导致车辆需求的结果差别很大，而高度动态的供需环境，给真实场景模型训练带来重大挑战，其关乎该城市当地车辆供需量间隙大小，并且该测试实验的环境也是一个很大的挑战性，所以在未来将研究中，秦志伟将深入研究如何制定一个快速适应动态环境的鲁棒策略以满足网约车交易市场需求作为本场主题演讲的结尾。

参考文献：

[1] Xu, Z, Ll, Z, Guan, Q, Zhang, D, L, Q, Nan, J, Lu, C, Bian, w. and Ye, J, 2018, July. Large-Scale Order Dispatch in On-Demand Ride-Sharing Platforms: A Learning and Planning Approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Dota Mining (pp 905913). ACM.

[2] Tang X, Qin Z, Zhang F, et al. A Deep Value-network Based Approach for Multi-Driver Order Dispatching[C]. knowledge discovery and data mining, 2019: 1780-1790.

[3] Huang T , Ma Y , Qin Z T , et al. Origin-destination Flow Prediction with Vehicle Trajectory Data and Semi-supervised Recurrent Neural Network[C]// 2019 IEEE International Conference on Big Data (Big Data). IEEE, 2019.

[4] Qin Z T , Tang X , Jiao Y , et al. Deep Reinforcement Learning for Ride-sharing Dispatching and Repositioning[C]// Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19. 2019.

南京大学教授俞扬：更好的环境模型，更好的强化学习

整理 | AI 科技评论

第二届北京智源大会“强化学习”专题论坛上，南京大学人工智能学院俞扬教授做了《更好的环境模型，更好的强化学习》的报告。

强化学习到底能不能用？有些观点认为它不能用，这里不能用主要指的是它在真实环境下很难用，但是在类似游戏这种封闭环境下强化学习其实都是很好用的，而且能够达到很好的效果。

我们该如何把强化学习用在真实的环境里面呢？针对于此，俞扬在这次分享中带来一个很简单很直接的办法：使用模拟器。模拟器就是强化学习所运行的环境，通过它我们就可以把强化学习跑出来并得到一个很好的效果。用一句话来说就是，真实环境不好用，模拟器来凑！

俞扬的团队已经和众多企业合作把基于数据驱动模拟器的强化学习技术用在了淘宝搜索、在线购物、滴滴出租车、仓库派单、砍价机器人等业务中并取得了很好的效果。另外基于数据驱动模拟器的强化学习技术途径，正在孵化南栖仙策公司，在制造、物流、营销等场景落地赋能，致力于将人工智能决策的力量转化为生产力。

在报告最后，俞扬总结了在真实场景下做决策这件事情的四个层次：1. 人直接来做决策；2. 人为设定模拟器；3. 用预测的方法来替代决策；4. 数据驱动的模拟器。

以下为演讲正文：

今天这个报告主要是集中在强化学习里面的环境模型上面，我们经常说的 model 指的是环境，像监督学习里面，我们经常说的 model 可能指的是神经网络的模型或者是其他的模型，像 model based reinforcement 指的就是环境模型。

一、强化学习 VS 监督学习

这里简单介绍一下强化学习和机器学习的关系，最经典的一个机器学习领域的划分包括： 监督学习、无监督学习和强化学习。

这种划分基于对某一任务上面的不同反馈机制，监督学习是立即反馈，无监督学习是没有反馈，强化学习就是涉及他们两者之间的一种滞后的反馈。这种滞后的反馈对应的就是做决策这样的任务： 做完决策过一段时间后才能看得到最终的决策结果是什么。

强化学习其实就是用来处理这样的决策任务，如果把它和监督学习做预测进行对比，那么我们主要回答的是怎么做，怎么做才能达到我们的目标，不同的任务就会导致我们的整个算法流程会有很大的不一样。

比如说在监督学习里面，做预测主要是解决数据里面是什么，做决策主要想回答怎么做能够达到我们的目标以

使得我们的学习过程变得不一样。如监督学习会从环境里采集一些数据然后给数据打上标记。输入和输出都有了之后我们把中间对应的函数用监督学习的算法还原出来，此时这个可用模型指的就是做预测的输入到输出的模型。

强化学习模型则不一样，因为强化学习要考虑的是有没有达到想要设定的目标。我们从环境中通过我们自己的模型采一批我们自己的策略和一批数据以后，我们学好策略后又会回到这个环境中，我们去验证一下这个策略有没有更好、有没有达到我们的目标。所以我们从目标上面最容易看得清楚这件事情的过程是和监督学习不一样的。

监督学习通常是针对一个固定的数据分布来优化损失，但是在强化学习里面，我们的数据分布是 D πθ，θ就是我们自己的决策、我们自己的策略。也就是说当我们的策略发生变化时我们的数据也会发生变化，这样的一个性质就会造成多变。一个很重要的区别就是，监督学习是在一个固定的分布上面来优化损失，这里假定我们过去的数据和我们将来要应用的数据来自于同一个分布，这样学出来的东西就是可用的。

在强化学习里面做决策时会碰到什么样的数据，实际上取决于做的决策是什么样的，这就使得我们很难在历史数据上面训练得到一个比较好的策略。比如说下围棋，我们选择的不同走法会把我们带到不同的未来，那么要找到最好的未来光靠一个固定的数据集就不太可行，比如说在阿法狗的第一个版本里面用了到人类下棋的数据集，但是在后面就没有用到。

我们光从一个固定的数据集上很难去找到一个最优的决策，那么如果我们不能从数据上面直接找到最优的决策，我们应该怎么做呢?

实际上就是要靠试错，也就是说要试试看，今天走到这一步拿到了 80 分，那么走到另外一步会拿到多少分？我们去尝试一下，看到了这个结果，才能知道拿到多少分，这样才能找到比较好的一个决策。

有这样一个区别以后，虽然我们看到强化学习也是通过大量的数据在学，但是这个数据是我们所谓的 agent，也就是我们在环境里面主动采样出来的策略。所以如果说监督学习依赖的原料是数据集，那么强化学习依赖的一个原料其实是它学习的环境。所以这个环境的存在与否直接决定了我们的强化学习能不能高效的运行起来。

那么我们现在能看到的强化学习用的比较好的案例，比如说下围棋打游戏这样的案例，其环境都是已经给定的。那里面的所有规则都是非常清楚的，所以我们可以在计算机的环境里面进行大量的采样，那么我们可以取得达到人类专家、甚至是能够超越人类水平的效果。所以我们可以看到，当强化学习能够取得这样好的结果时，它所运行的环境是一个我们已经非常清楚的或者是一个封闭的环境。

如果想把它用在一个真实业务上面，我们就会面临很大的问题。首先需要强调的是很多业务问题、很多真实的应用都是需要做决策的。如果我们确实能够把现在做决策的这种能力放到各种应用中去，那么就能够比人做的好、比专家做的还好，那么我们自然会有很大的收益。但是如果我们去看具体的这些应用时，我们会发现它和打游戏很不一样的地方就是我们没有一个现成的做好的一个环境，在真实业务中的环境都是一些真实、开放的、一些边界不那么确定的一些环境。

那么我们能不能直接把强化学习用在这样的环境里面呢？实际上如果我们去看任何一个强化学习运行的过程，我们都会发现它需要很多的探索。所谓的探索就是我们要去试错，所以这也引来了前段时间对于强化学习能不能用的这么一些批评，有些观点认为它不能用，其实不能用主要指的是它在真实的环境里面很难用，但是在游戏这种封闭环境下面其实都是很好用的，而且能够达到很好的效果。

二、模拟器

针对这样的情况，我们应该想些什么办法能够把强化学习用在真实的环境里面呢。其实有一个很简单很直接的一个想法，那就是使用模拟器，模拟器就是强化学习所运行的环境，通过它我们就可以把强化学习跑出来并得到一个很好的效果。

那么如何得到这样一个模拟器，如何让我们的模拟器能够和我们的真实环境对应上？实际上这并不是一个很新鲜的事情。我们可以看到在制造业和工业界里面有大量的模拟器的工作，所以以往的模拟器大部分是基于人类的知识来进行构造的，也就是用人工来构造的。里面包括比较常见的机器人控制、流体动力学、物流： 我们可以把物流的整个场景虚拟的建模出来； 上面这些已经有很多人研究了很多年，其实就是为了把里面的动力学模拟做的又快又准。

虽然在很多场景上面可以做出来模拟器，但是它们精度还是存在一些不足的，特别是当我们的场景稍微有点复杂的时候，我们很难完全准确的去模拟这个场景。那么当它出现误差的时候，我们能不能使用这样的模拟器呢？实际上是可以的。现在有一些技术能够让我们基于这些人工构造的模拟器做出一些比较好的决策出来。

举一个例子，这是我们之前做过的一个关于机械手臂的控制。在真实环境下我们可能不太清楚机械手臂具体有多长，但是由于我们有一个类似的人造的环境，我们可以通过这个环境产生大量的不同机械手臂的长度，使得我们在不同的环境中训练我们的 agent 以用来适应这样的环境。所以我们在真实环境中学出来的模型可以对环境做少量的试探，做完试探后的结果可以用来刻画这个环境的样子，它就可以很快地把经验用起来去适应环境。

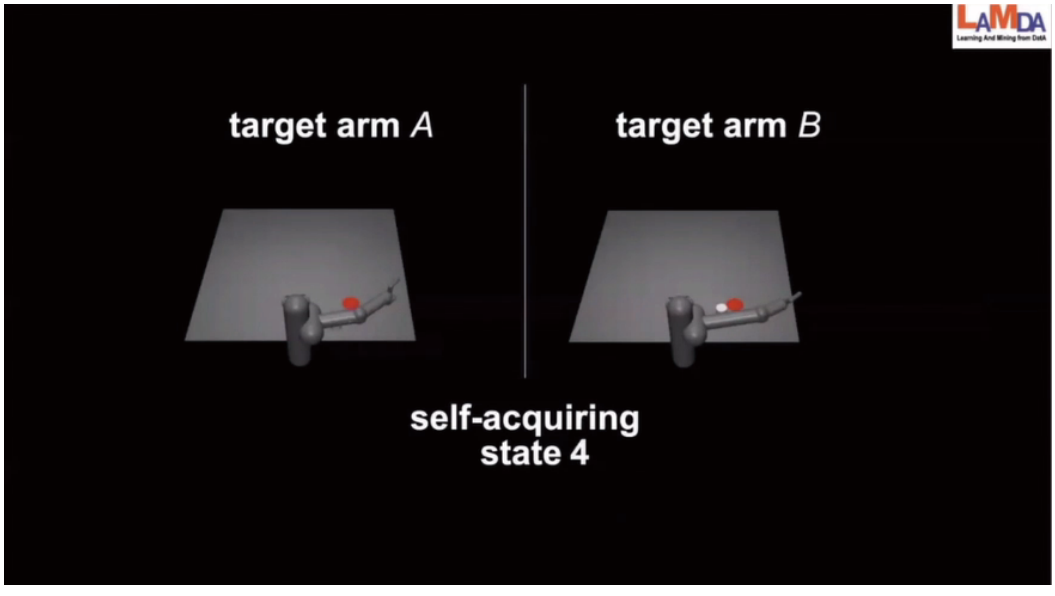


图 1： 模型最初并不知道机械臂有多长，但做几个动作之后便知道了。

比如说上图左右是两个不同的手臂，刚才执行的 5 个动作就是在试探这个手臂到底是多长，实际上它并不知道手臂有多长，那么做了 5 个动作的适应以后就可以完成这个问题了。也就是我们把经验的模型重用起来来完成这个任务。可以看到，使用这样的方法之后，我们在环境里面做少量的几个试探后就可以一定程度的完成任务。 但是如果我们从头来训练这个任务，可能需要上百万的试错才能完成。所以这是一种能够利用人工的模拟器方式。

另外虽然人工的模拟器不是很准确，但是也能够给我们的学习进行加速。之前我们做过这样一个工作： 在新机的环境下把各种强化学习的技术整合起来，来看我们能够把这个问题解决的多快。之前在 19 年的工作中，我们做过在训练两天后，能在一台计算机上面打败它的内置 AI 的这样一个实验。

其实我们可以做一个对于星际争霸游戏这样复杂的任务： 我们可以把它抽象，先做一个人工的小的模拟器，里面只有几条简单的规则。通过这个模拟器进行预训练以后，我们再把它搬到复杂的大的游戏上面来。这样的一个做法可以使得我们在任何一台计算能力更弱的机器上面做到在一个小时之内学会打败他的内置 AI，只需要几个迭代，就可以达到一个非常高的胜利。这也说明人工设计的模拟器实际上是有很大用处的。

但是同时我们还在想，如果模拟器一直是人工设置的，那它会面临很多问题。第一个就是在人工设置的过程中人力开销是很大的，另外一个就是人工设置的模拟器都定下来以后不能很好地跟随环境的变化而变化。所以如果我们能够从数据上面还原模拟器，就可以缓解这两个问题。

从数据上面去做模拟器就是在当前的状态下做一个决策，然后确定下面会进到哪个状态去，这个则是模拟器希望给出的一个答案。我们看到这个也是一个输入输出的模型结构。而且如果我们有一些历史的数据，我们历史上看到了哪些状态、做了哪些决策以后，它随之进入到了哪些新的状态去，这个看起来很自然的是一个监督学习任务。

那么是不是可以用监督学习来帮我们把模拟器建好？实际上在一些情况下面是可以的，比如说这个是去年一个做机器人的工作。这个机器人的实体控制靠收集的大量数据，然后从它的数据上去做一个监督学习，来把这个模型还原出来，然后再在还原出来的模拟器里面学会怎么去控制机器人。结果它的控制会变得像生物一样更自然一些。

但是这种用法实际上有一个条件： 机器人在实验室的环境下能够完全掌控。换句话说，我们在实验室的环境下能够收集到大量的它在各种状态下面做出控制决策后的响应，这样的数据我们是能够收集得到的。收集到这样的数据以后，我们可以做一个监督学习来还原出一个模拟器出来。

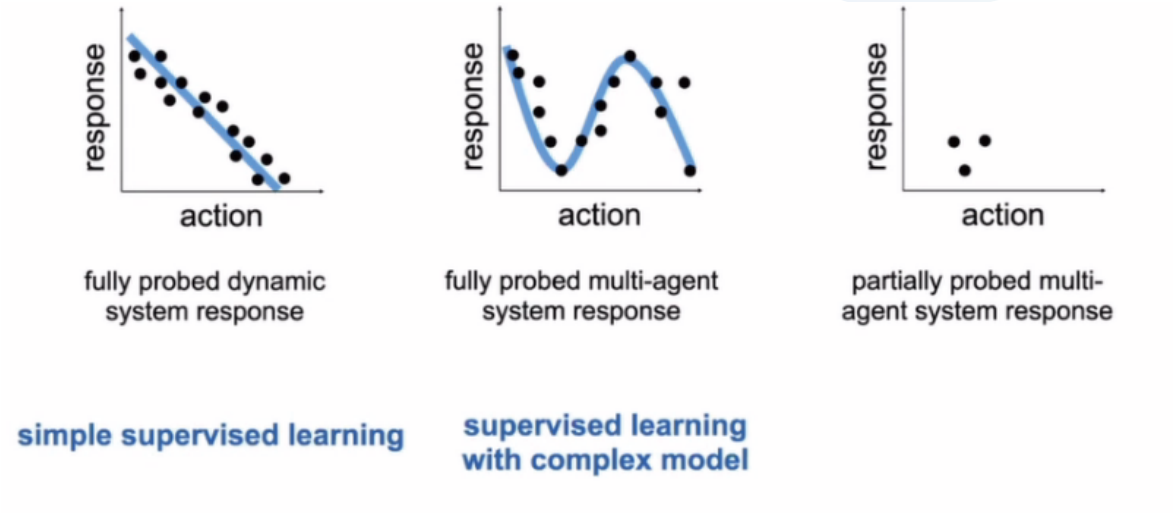


图 2：关键的问题

虽然更多的时候我们面对的数据会更复杂，但是如果所有的决策空间都被我们遍历过，拿到这样的数据之后，模拟器也是可以学出来的，可能只是需要一些更复杂的神经网络模型而已。

其实我们在很多实际应用的环境下做过的决策很少，我们不可能把所有的角色全部遍历过。那么在这样的情况下面，我们还有没有可能把模拟器建好，这个是我们面临的一个最关键的挑战。在缺少大量的响应数据和它的决策情况下，我们有没有可能做好?

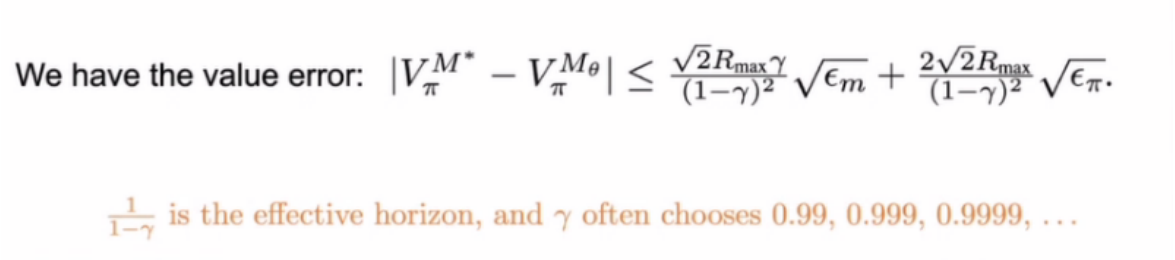
在这一方面我们可以先来看一下用简单的监督学习来做会有什么样的问题。我们假设真实的环境就是 M*，我们会有历史上的一个采样策略，我们历史上去做一些决策，然后能够收集到一些数据。

我们把我们历史上做了哪些状态、有了哪些状态、做了哪些决策作为我们的输入，然后把它下面变到哪些状态作为我们的标记，那么我们可以简单的训练一个监督学习。比如说我们可以用 KL 作为我们的目标来训练监督学习的一个模型，那么这样我们构造出来的模拟器会怎么样呢?

If we have environment model error: $\mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}}\left[D_{\mathrm{KL}}\left(M^*(\cdot|s,a), M_\theta(\cdot|s,a)\right)\right] \leq \epsilon_m$,

and bounded policy divergence: $\max_s D_{\mathrm{KL}}\left(\pi(\cdot|s), \pi_D(\cdot|s)\right) \leq \epsilon_\pi$,

实际上从我们最新的结果可以看得到，当我们构建的模型在 KL 上面有一个损失有一个界限以后，可以看得到它最后的结果。也就是说我们在模拟器里面去评估一个策略和在真实的环境下面去评估一个策略，它们两个之间的差别有多大，就是下面这个式子所展现出来的。



那么这里特别想强调的一点，关于 1− γ这一项是什么意思？γ实际上就是我们在学强化学习的时候每一步会有的一个γ的折扣，折扣通常会选像 0.99 或者 0.999 这样的一个比较接近于 1 的数字。那么 1/（1− γ）实际上就是全部的长度加起来以后，它的折扣的权重一共能有多大，所以我们也把它叫做 effective horizon。它是我们做决策能够看多远的一个评价。

在公式右边我们可以看到是 effective horizon 的平方，这实际上是一个很大的数，如果我们用这样一个方法去学一个模拟器出来，我们面临的问题就是当我们多走了几步以后，它的误差会迅速的放大，会导致我们最后学出来模型的策略很难用。

有没有办法能够缓解这件事情呢？实际上我们可以把这种简单的监督学习换成对于我们分布的一个匹配。这里的分布指的是我们一直在环境里面去运行的策略，我们会不断的收集数据，我们一直运行下去以后，我们收集到这个数据会呈现一个什么样的分布？

如果只看状态那就是上图第一行的定态分布，如果看状态和动作的，就是第二行的样子，如果再把这个模型的某一步的转移概率加上去就是 state−action−next−state 分布。所以这里我们发现我们其实可以把我们的目标从简单的监督学习换成分布的匹配，主要是做后面这一个匹配也就是 state−action−next−state 分布匹配。

那么如果换成这样的一个匹配以后，我们的算法应该怎么实现呢，之前我们有一些论文的发表，实际上也涉及到了这个方面的一种新的损失，一种新的目标的使用。总的来说，从 idea 上来说，实际上是可以通过对抗学习的方法来使我们的分布匹配（分布匹配）做的比较好，那么如果我们换成这样的一个目标以后，我们能得到一个什么结果？

上面的设置和前面简单的监督学习是一样的，我们能得到的结果就是我们去掉了其中的γ一项的平方，就会使得我们这一项的误差降低 1000 倍以上。那么这个是我们这个理论上面的一点进步，实际上还有很多关于模型学习的问题没能得到完全的回答。但是我们在算法应用上面实际上已经做了一些尝试，所以下面我们就想做一些抛砖引玉，然后看看我们之前实践过的一些环境上面做出来的一些应用，希望能够给大家一些启发。

三、应用

1. 淘宝搜索

这个地方要做的决策是什么呢？就是来了一个用户的搜索请求以后，我们需要展示一个商品以及商品的顺序。以往的做法大多数是以预测的方法来做的，也就是说我们过去哪些人点击了哪些东西，然后我们总的来说是把这样的一个数据作为一个监督学习的方式来进行利用，然后把学到的模型拿到今天的场景下面来做推荐。

这里我们考虑到，对用户而言，实际上以什么样的顺序去呈现这个商品会带来很大的影响。我们想做决策，但是又不想去干扰真正的用户，所以我们这里的一个路径就是用历史上用户的数据来学一个模拟器出来。这个模拟器里面是虚拟的用户，所以我们在模拟器里面推荐一些商品以及商品的顺序出来，然后虚拟的用户就会来买，会点击或者是跳过去，那么我们就可以在模拟器上面去寻找怎样的方式能够有更好的销售，能够让用户有更好的体验。

之前我们做了实验的效果，在模拟器里面我们的 GMV 提高了 4 个点。然后我们就想在模拟器里面做的结果能不能拿到真实的环境下面直接去试试看，因为这是我们做的第一个应用例子的环境。

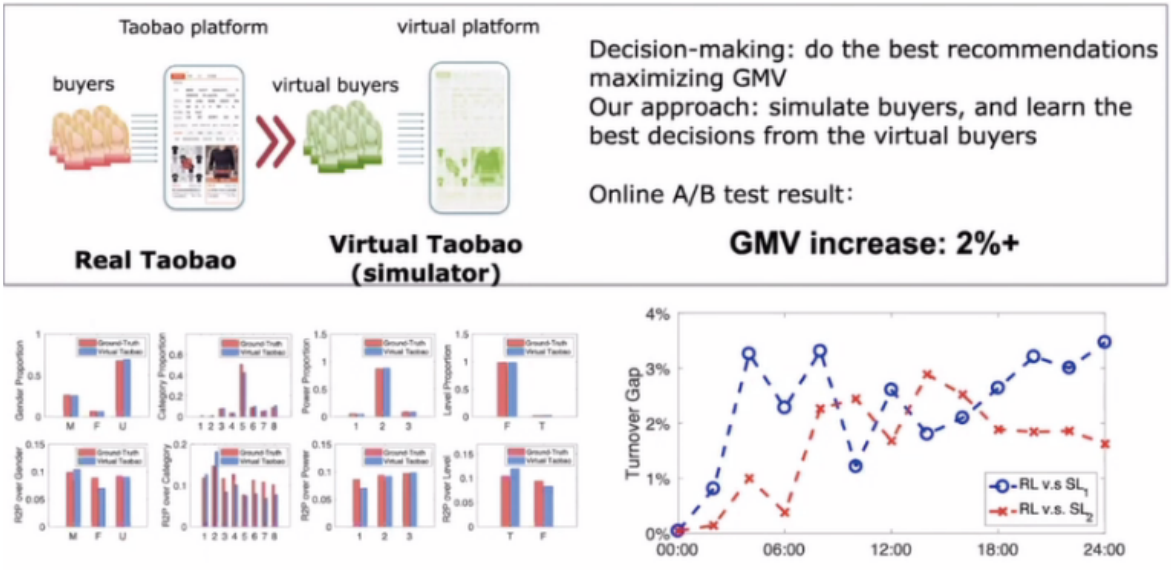


图 3：案例一，用户模拟

实际上在上图右下角显示的就是真实的淘宝数据分布和我们的模拟器里面生成的数据分布，红色的是真实的，蓝色的是模拟器的，可以看到它们已经比较接近了。所以我们当时也是把模拟器里面训练出来的决策的模型，比如说怎么去做推荐的模型，直接放到了真实的环境下面去做测试。AB 测试的结果显示处比最好的 baseline 提高了两个百分点。所以这个可以看得到我们即使是没有做任何的迁移、没有做任何的环境的扰动这些事情就直接把这个模型拿出来，也是在一定程度上是可以用的。

2. 在线购物

这个例子实际上和淘宝是比较像的，它也是一个在线购物的环境。但它要做的事情是在另外一个侧面，也就是怎么样去设计这个商品才能够更好地去满足我们的消费者。这里以冰箱举例，冰箱有很多属性，消费者可能更喜欢哪些属性呢?

我们的做法其实是一样的，我们把这个买家虚拟出来，然后我们去调整商品的属性，我们可以看到下面是一个实验的结果，橙色的柱子是现有的在数据上面的商品的一个分布，蓝色的是我们在模拟器里面寻找到的更好的一个分布。

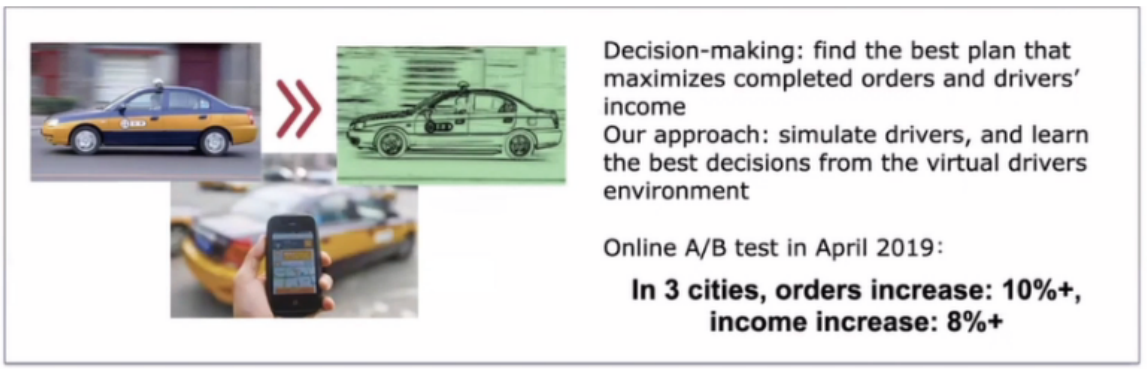
图 4：案例二，市场

比如说我们看第一个图上面，它会预测现在卖双开门的冰箱比较多，但是实际上我们的模拟器里面求解出来的结果认为喜欢的用户可能不是那么多，反而是对开门的冰箱可能会卖得更好。

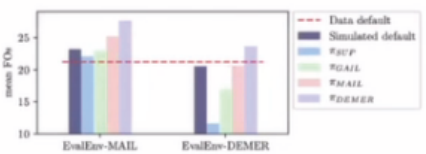
这个例子就是在右上角画的这两个冰箱。

3. 滴滴出租车

这里要做的事情是如何给司机推荐一个他的 program 和 plan，然后能够使得司机的收入尽可能的增加，这里我们的做法也是类似的。



data test



A/B test:

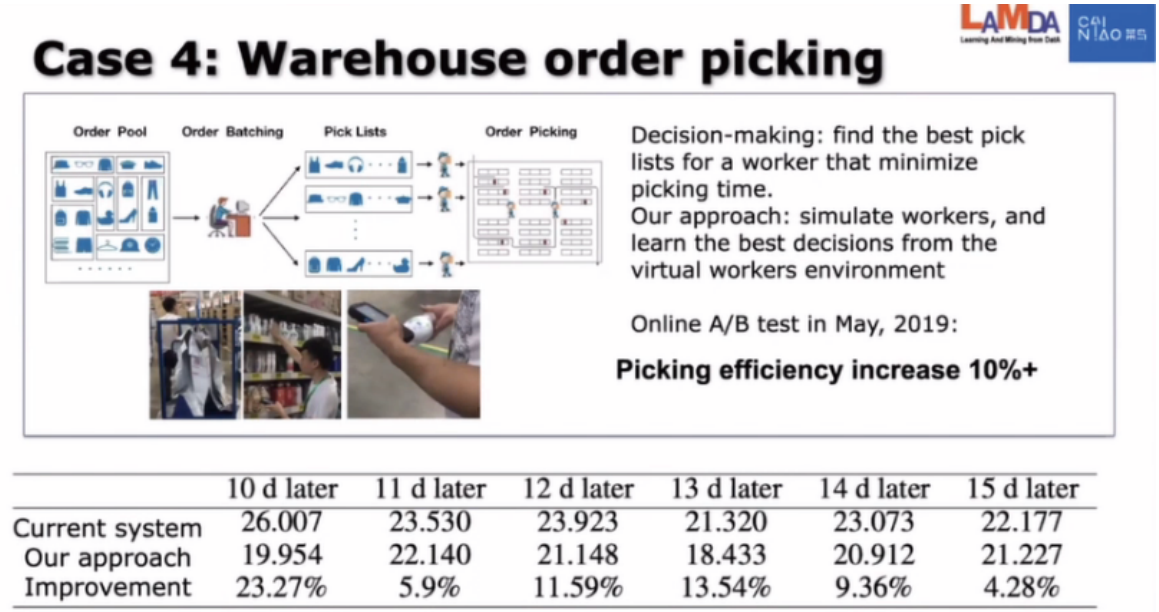
Cities	ΔFOs(%)	ΔTDIs(%)
City A	+10.73	+6.16
City B	+10.16	+9.38
City C	+18.47	+17.84
Total	+11.74	+8.71

图 5：案例三，出租车实际模拟

我们从历史的数据里面去把司机的虚拟模型学出来，这就是我们的模拟器。然后我们在模拟器里面给司机安排一些程序使得司机的收入能够提高。我们在去年 4 月份的时候完成了这个事情的线上测试，在线下的我们自己测试的时候差不多提高了 13 个点左右的完单量，然后在线上测试实际上提高了 11 个点多的完单量，同时还提高了司机的 8% 的这么一个收入，这也是一个有意义的应用环境。

4. 仓库派单

在这个仓库里面，我们实际上要做的是在一个仓库里面进行一个派单的这样一件事，这是很小的一个环节。在这个环节我们要把很多订单给一个工人，工人拿订单去把我们下单的商品放到袋子里面去，那么把哪些订单给同一个工人去拣货，这就是一个优化的问题。



	10 d later	11 d later	12 d later	13 d later	14 d later	15 d later
Current system	26.007	23.530	23.923	21.320	23.073	22.177
Our approach	19.954	22.140	21.148	18.433	20.912	21.227
Improvement	23.27%	5.9%	11.59%	13.54%	9.36%	4.28%

图 6：案例四，仓库订单选择

以往在物流场景上面的做法通常是运筹学这种传统的做法，它首先需要去制定目标，这个目标怎么制定呢？以往的做法就是我们去把地图给拉出来，然后我们看每一个商品在地图上放在什么位置，然后我们就按照这个商品在地图上的位置规划出一个最短的路径来作为指标给工人进行派单。但是如果是针对人造的地图来进行这么一个派单，有很多细节其实没有办法体现进去。

比如说工人拿货的时候有不同的高度，这个货物有不同的重量和不同的形状，这些实际上都会影响他拣货的效率。如果我们只是看地图的路径长度，则就会忽略掉这些因素。所以用我们的方法来做的时候，也是从数据上面去把这个功能给还原出来，所以我们就会有虚拟的功能，然后我们就给虚拟的工人来派单，然后看这个虚拟功能它怎么派会完成的时间会最少。这个事情做完了以后，我们去年 5 月份的时候在这个仓库里面做了对比实验。这个对比就是现有的系统派一单、我们的系统派一单，对比的 6 天下来，我们可以看到上面的数字显示的是：针对每一个商品，工人去拣货平均下来的时间是多少秒？6 天平均完以后，我们可以看到我们的效率提升了 10% 以上。

5. 砍价机器人

有人买东西的时候经常会砍价，那么我们能不能自己做一个机器人出来和那个人来砍价，目标可能是希望我们的这个二手货平台的成交率要尽可能的高。

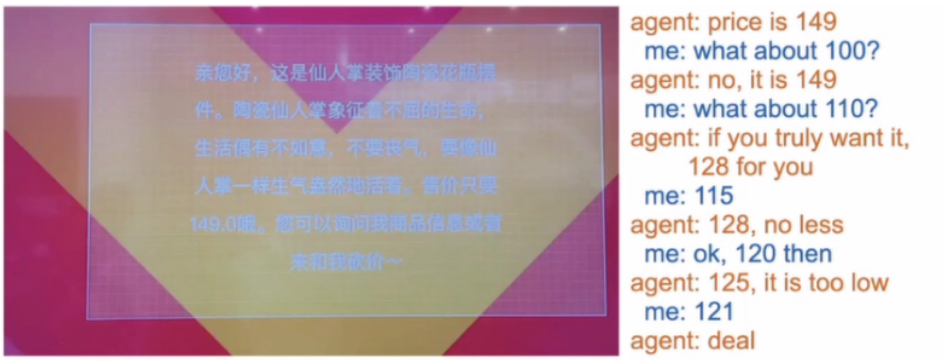


图 7：案例五，砍价机器人

这个是当时在云栖大会上的一个展示，这里面还涉及到一些别的像自然语言处理不是我们做的，我们做的是里面的砍价策略。当对方出了一个价之后，我们要决定是卖了还是说我们要还一点价。所以我们这个实验做完了以后，我们和人去比砍价，要比历史数据上面的成交率提高一倍。

我们在历史数据上面去学出来一个买家，然后这个买家来和我们砍价，所以在模拟器里面就有很多买家来和我们砍价，然后我们就和买家来学习我们怎么砍价使它的成交率会更高。

四、总结

以上的这几个案例希望能够抛砖引玉、给大家一些启发。另外我们在做了这些实践以后，现在做一个总结，看看对于在真实场景下面做决策这件事情，我们都有哪些做法。

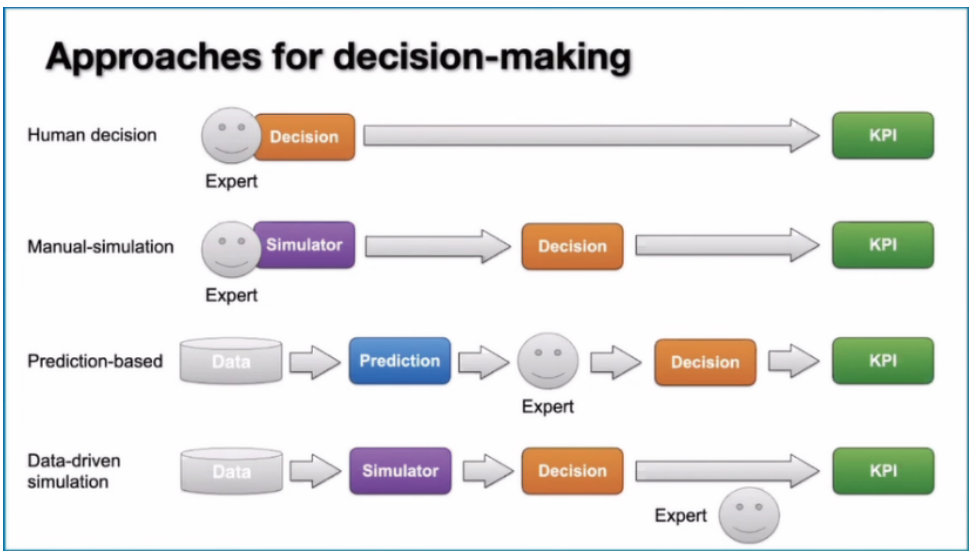


图 8：决策方法

1. **人直接来做决策。**人们亲自来做决策来使目标最优化，这可能是最原始的做法。

2. **人为设定模拟器。**在模拟器里面去求解最优的决策，然后把它放到真实的环境去用。这样的好处就在于我们有了模拟器，我们可以用很多算法来求解最优的决策。

3. **用预测的方法来替代决策。**比如说以前做推荐系统基本上是从数据里面来做一个预测，然后将预测的结果交给人或者人做的一个规则，以此来把它变成一个决策的结果。举个例子就是我们在做推荐的时候通常会有一个预测模型，然后它能够告诉我们什某样商品购买率可能是多少，然后我们就按照购买率从高到低来把它展现出来，把商品展现出来。

实际上在真实的系统里面已经看到从高到低这个规则已经不是一个最优的规则了。实际上我们在用预测来解决决策问题的时候，中间都有一个转换。但是如果我们把第二种方式和第三种方式进行对比就会发现第三种方式很好的一个地方就在于它是从数据出发的，当环境发生变化时，它能够适应得上，但是它的缺陷就在于他没有办法去寻优，没有办法自己去寻优，因为它没有一个模拟的环境。

4. **数据驱动的模拟器。**我们首先具有从数据出发这样的一个优势，我们可以根据数据来适应环境。另外我们可以在模拟器里面去寻找最优的决策，所以这一条路线有可能实现人不在回路里面。

上面的前三种方法都是人在环境中、在整个过程中起到关键的作用。而第四种方法人则是可以在旁边，这不代表人不做监管，所有的决策当然都是需要符合我们设计者的目标，但是它可以完全自动运行，这也是我们现在希望能够完全实现的一条路径。

上交大张伟楠：基于模型的强化学习算法，基本原理以及前沿进展

整理：智源社区 张文圣

在第二届北京智源大会“强化学习”专题论坛上，上海交通大学张伟楠副教授做了主题为《Model-based Reinforcement Learning: Fundamentals and Advances》的演讲。

在本次报告中，张伟楠教授介绍了基于模型的深度强化学习算法的研究背景及最新的前沿进展。张教授从无模型强化学习与有模型强化学习的对比开始，结合基于黑盒的有模型强化学习的发展历史，深入浅出地讲解了有模型强化学习诸多算法的基本概念、算法起源、实现原理、理论分析以及实验结果等，详细介绍了所在课题组在这一领域的最新工作进展，并对这一领域今后的发展方向进行了前瞻性的总结概述。我们查阅演讲中介绍的相应算法的论文原著以及相关文献，对本报告进行了一定的补充说明。

报告分为如下几个部分：

1. 基于模型的深度强化学习算法研究背景。
2. 基于黑盒模型的 Dyna 算法。
3. Shooting methods：基于随机采样的 MPC 算法。
4. MBPO 算法理论分析与实现方法。
5. 最新的 BMPO 算法理论分析与实现方法。
6. 总结。

一、基于模型的深度强化学习算法研究背景

深度强化学习算法自提出以来，常用于 Atari Game、围棋、DOTA、星际等虚拟场景中。人们通常将这种不建立环境模型，仅依靠实际环境的采样数据进行训练学习的强化学习算法称为无模型强化学习（Model-Free Reinforcement Learning，MFRL）算法，也即是不依赖于环境模型的强化学习算法。这种方法适合应用于深度神经网络的框架，人们将大量数据以 mini-batch 的形式传入神经网络，可以对价值网络或者策略网络进行非常高效的训练。

然而，MFRL 发展中遇到的一个困境：数据采集效率（sample efficiency）太低。在有监督或无监督学习中，人们构建一个目标函数，通过梯度下降（或上升）的方式，不断趋近理想结果。与有监督 / 无监督学习不同的是，强化学习属于一种试错的学习范式，当前策略的采样结果如果无法有效帮助当前策略进行提升，则可以认为当前试错的采样结果是无效采样。在 MFRL 训练过程中，智能体有大量的交互采样属于无效采样，这些采样没有对行动策略的改进产生明显的影响。为了解决无模型强化学习中的这一数据效率低下的问题，人们开始转向基于模型强化学习（Model-Based Reinforcement Learning，MBRL）的方法。

MBRL 的基本思想在于首先建立一个环境的动态模型，然后在建立的环境模型中训练智能体的行动策略，通过这种方式，实现数据效率的提升。

将 MBRL 与 MFRL 对比来看，MBRL 存在如下特点：

1. 环境模型一旦建立起来，便可以采用 on-policy 的训练方法，利用当前采样得到的数据训练当前的策略，在这种情形下，采样效率是最高的。
2. 建立环境模型后，便可以选择性地不再与实际场景交互，在模型中进行训练学习，完成训练后再在实际场景中投入使用（off-line RL，也称为 batch RL）。
3. 相比于 MFRL，MBRL 数据采样效率会往往有较大的提升。
4. 存在模型与实际环境之间的复合误差问题（compounding error），模型向后推演的幅度越长，推演误差就会越大，直至模型完全失去作用。

MFRL 存在如下特点：

1. 相比于 MBRL，MFRL 拥有最好的渐进性能（asymptotic performance），当策略与环境交互达到收敛状态时，相比于 MBRL，MFRL 下训练所得策略所最终达到的性能会更好，能够避免出现复合误差的问题，因而在实际环境中表现会更为优异。
2. MFRL 非常适合使用深度学习框架去采样超大规模的数据，并进行网络训练。
3. MFRL 经常采用 off-policy 训练方法，这种情况下会有偏差（bias）导致的训练效果不稳定（instability）的问题。
4. MFRL 需要进行超大量的数据采样，因而需要超高的算力要求，这种算力要求是很多科研院所或者企业所无法负担的。

关于 MBRL 的进一步分类，其主要包括黑盒模型与白盒模型两类：

黑盒模型中，环境模型的构造是未知的，仅作为数据的采样来源。由于采样数据来自于黑盒模型，而不是和真实环境交互得到，因此这些来自模型的采样数据不计入数据采样效率的计算中。虽然从计算结果来看 MFBL 的数据采样效率较高，但由于训练过程中使用了大量基于模型采样的数据，因此从采样数据总量上来看，实际采样了更多的数据。常用的基于黑盒模型的 MBRL 算法包括 Dyna-Q、MPC、MBPO 等。

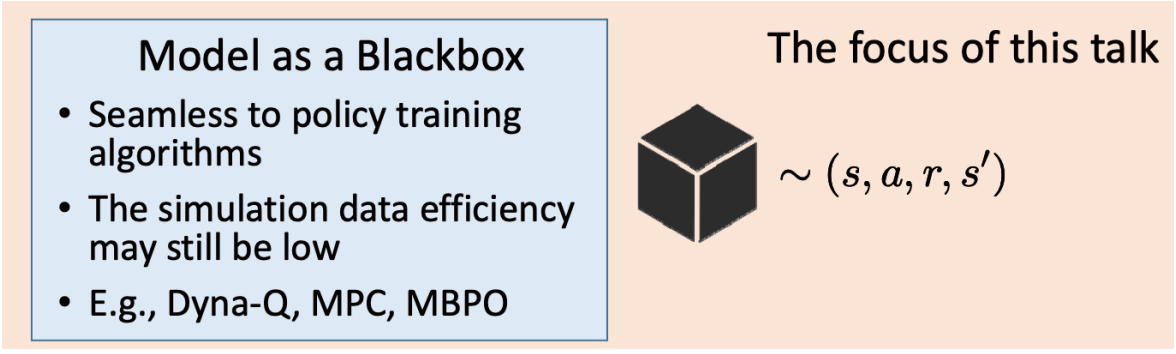


图 1：黑盒模型

白盒模型中，环境模型的构造是已知的，可以将模型中状态的价值函数直接对策略的参数进行求导，从而实现

对策略的更新。常用的基于白盒模型的 MBRL 算法包括 MAAC、SVG、PILCO 等。

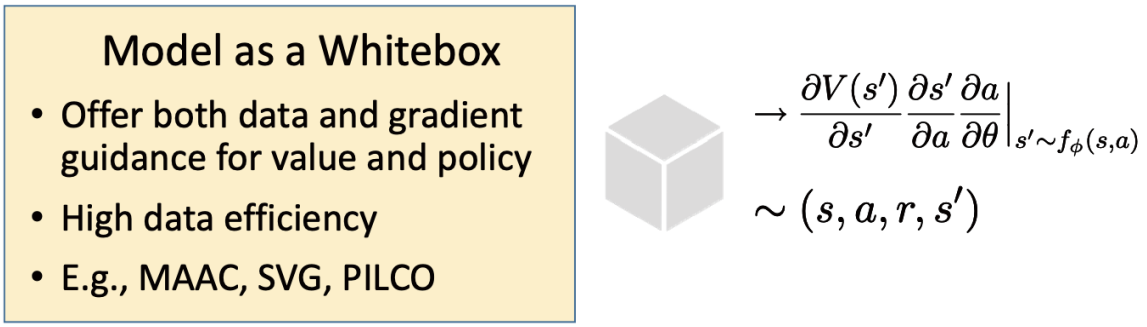


图 2：白盒模型

二、基于黑盒模型的 Dyna 算法

在报告中，张伟楠教授主要主要介绍了基于黑盒模型的 MBRL 算法。这里环境模型的构造未知，通过采集更多数据的方法来进行策略训练。这样处理的一大优势在于算法具有更好的普适性，可以和几乎所有无模型强化学习方法结合。

MBRL 算法主要流程如图 3 所示:

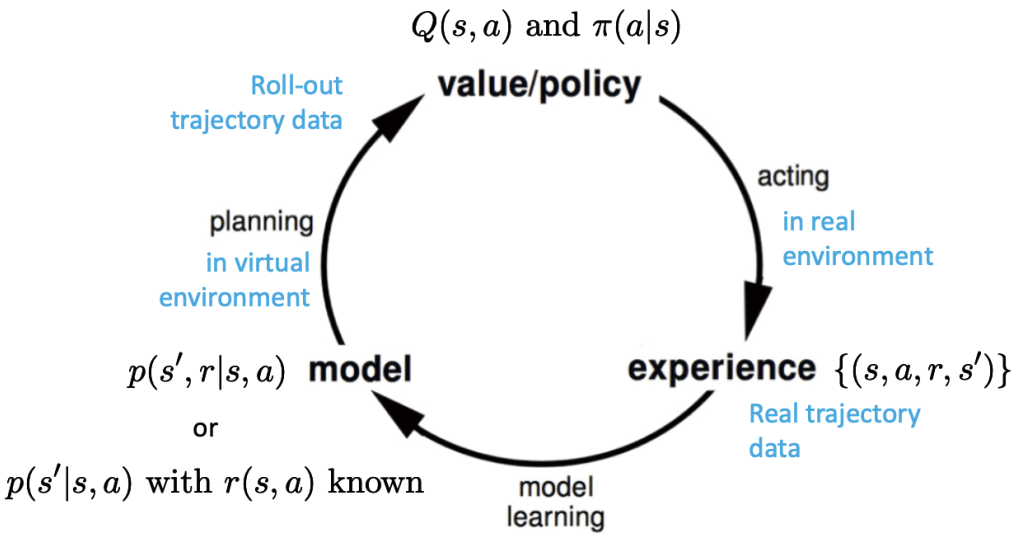


图 3：MBRL 算法主要流程

首先，当前的价值函数 Q(s, a) 以及策略函数π (a | s) 与真实环境进行交互，完成交互后采样出环境反馈的数据 experience{(s, a, r, s')}。然后通过采样出的数据来训练建立的环境模型 p(s' , r | s, a)，环境的模型本质上是通过输入的当前状态 state 以及采取的动作 action，来预测产生的 reward 以及下一步的状态 state，很多情况下

reward 是根据先验规则或领域知识生成，这时模型只预测下一步的状态 state 即可。接下来在当前模型中进行 plannning，也就是通过当前模型进行数据的采样，通过数据采样的结果去训练新一轮的价值函数 Q(s, a) 以及策略函数π (a | s)。

Q-Planning 是最简单的 MBRL 算法。通过与真实环境交互的数据来训练环境模型，然后基于当前的环境模型来进行一步 planning，从而完成 Q 函数的训练。首先采集智能体与环境交互的 state 以及采用的 action，将数据传入建立的黑盒模型中，采集并得到模型虚拟出来的 reward 以及下一步的 next_state，将传入模型的 state、action 以及模型虚拟出来的 reward、next_state 进行一步 Q-Learning 训练，这样就完成了一步 Q-Planning 算法的更新。期间智能体仅仅通过与环境模型交互来进行数据的采样，得到虚拟的 reward 以及下一步的 next_state，并进行策略的训练和更新，智能体未通过与实际环境交互数据来进行策略的更新。

Do forever:

1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(s)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

图 4：Q-Planning 算法

更进一步，可以将 Q-Planning 与 Q-Learning 结合在一起，并将这种算法称之为 Dyna 算法。Dyna 算法提出于 90 年代早期，完整的流程示意图如图 5 所示。可以看到，将中间的 direct RL 步骤去掉后，就是刚刚讨论的 Q-Planning 算法。

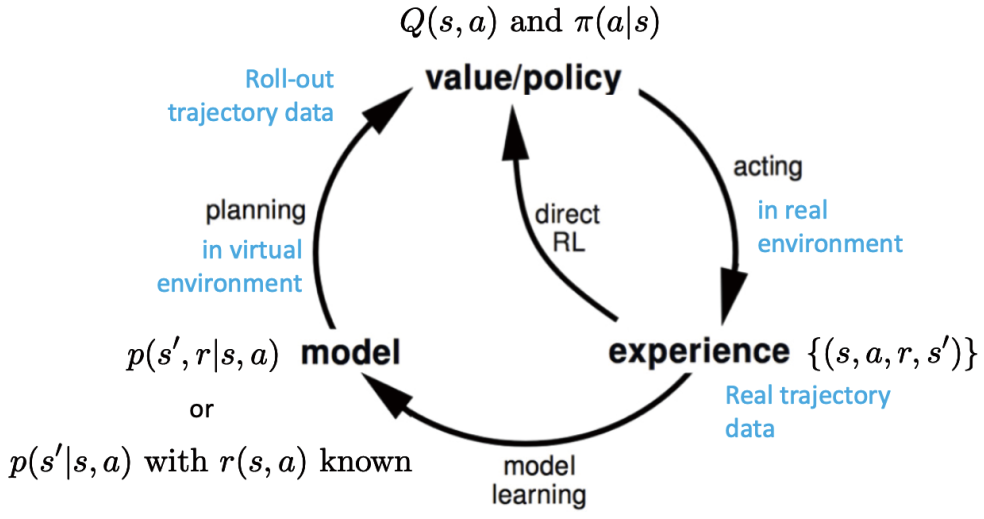


图 5：Dyna 算法流程示意图

在 Dyna 算法中，首先通过智能体与实际环境的交互进行一步正常的 Q-Learning 操作，然后通过与实际环境交互时使用的 state、action，传入环境模型中进行 Q-Planning 操作，真实环境中进行一步 Q-Learning 操作，对应环境模型中进行 n 步 Q-Planning 操作。实际环境中的采样与模型中的采样具有 1：n 的关系，通过这种方式来提升训练过程中的 sample efficiency。

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Execute action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

图 6：Dyna-Q 算法

以一个迷宫问题的场景为例，智能体从起始点 S 出发，前往目标点 G，智能体可以沿上下左右四个方向任意行动，期间不能穿过迷宫的四周墙壁以及深色方块所示的障碍物。

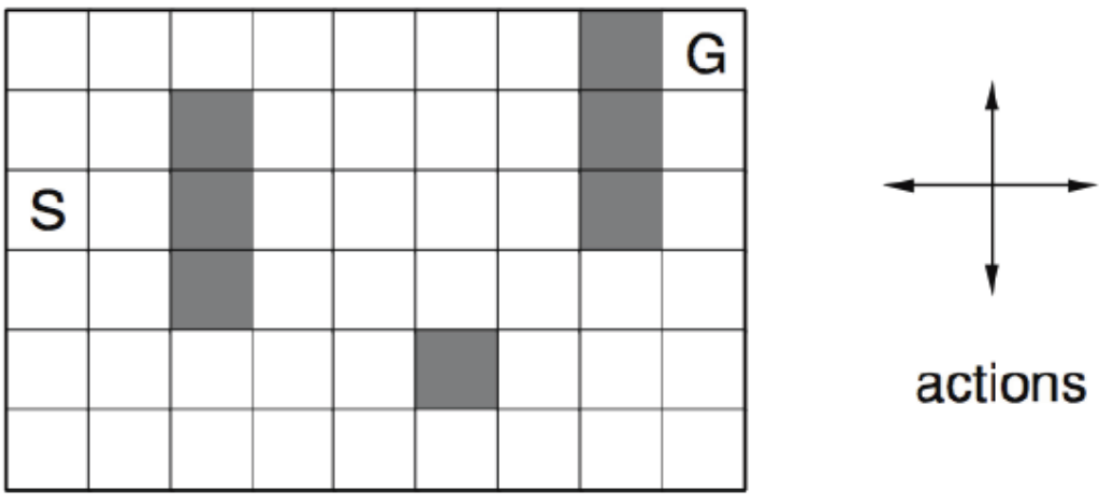


图 7：迷宫问题

下图分别对比了 0 planning–steps（只进行强化学习 Q–learning）、5 planning–steps（每做 1 步 Q–learning，做 5 步 Q–Planning）以及 50 planning–steps（每做 1 步 Q–learning，做 50 步 Q–Planning）的情况下的结果。这里的 n planning–steps 不代表前向（forward）推演 50 步，而是指从头进行一次 state、action 的采样，从而进行一步 Q–Planning，按照这种方式进行 n 次 Q–Planning 操作。如下图所示，横坐标为当前总计完成任务数量，也就是智能体在实际场景中累计从起始点 S 走到目标点 G 的次数。纵坐标为完成当前轮次任务时智能体从起始点 S 走到目标点 G 所使用的 action 步数，所使用的步数越低，则当前策略的性能越好。

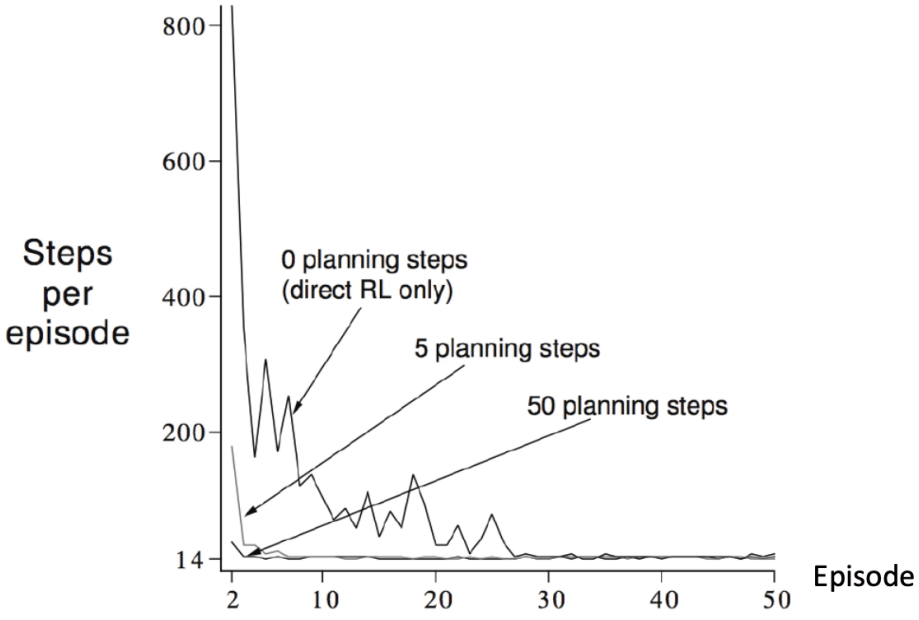


图 8：对比 0、5、50 Planning–Steps 的情况

从图中可以看到，当 n 值越大，环境模型推演越多时，所需的 episode 任务数量越少，policy 达到最优的时间点越靠前，数据采样效率越高。

MBRL 算法的发展面临着三个关键问题，也是发展的三个思路:

1. 环境模型的建立是否真的有助于 sample efficiency 的提高?
2. 所建立的模型基本都是基于神经网络建立，不可避免的会出现泛化性误差的问题，那么人们什么时候可以相信建立的模型并使用模型进行策略的训练与更新?
3. 如何适当地把握模型的使用尺度，来获得最好的或者至少获得正向的策略训练结果?

三、Shooting methods：基于随机采样的 MPC (Model Predictive Control) 算法

在已经获得一个有效的环境模型 p(s', r | s, a) 后，人们可以从初始状态出发，基于当前模型对后续动作进行序列采样 $[a_1, a_2, \ldots, a_T]$ 的方式，来获得一个采样序列 $[\{s_1^{(k)}, a_1, r_1^{(k)}, s_2^{(k)}, a_2, r_2^{(k)} \ldots\ldots s_T^{(k)}, a_T, r_T^{(k)}]\}_{k=1,\ldots,N}$，其中动作序列 $[a_1, a_2, \ldots, a_T]$ 通过完全随机采样的方式获得，与智能体的 state 等无关。对于当前状态为 s，a 的

智能体，可以建立一个动作序列 $\left[\mathrm{a}, a_1, a_2, \ldots, a_T\right]$，从当前的状态出发，在模型中连续向后进行 T 步推演，可以通过计算得到累计奖励。共计采样 N 次，可以得到 N 个积累奖励，对这 N 个积累奖励求取平均值，从而得到了在当前状态 s 下，采取动作 a 时，所获得的价值函数的无偏估计。

$$\hat{Q}(\mathrm{s},\mathrm{a})=\frac{1}{N}\sum_{k=1}^{N}\sum_{t=0}^{T}\gamma^t r_t^{(k)}$$

通过对每一个 action 的无偏估计的值的计算，可以选取价值函数的无偏估计值最大的 action，作为当前 state 下选择 action 的策略。

在以上整个过程中，几乎没有进行策略的学习训练，而是在基于环境模型的演绎树上进行搜索。当完成一步 action 的选择后，再在新的状态的基础上进行 MPC 操作即可。

模型建立之后可能会出现不确定性（uncertainty）的问题，一个解决思路是建立 N 个不同模型，每个模型都对下一步状态进行预测，将 N 个模型的预测结果取平均，作为接下来预测的结果（Probabilistic Ensembles with Trajectory Sampling，PETS）。

$$\text{Ensemble}\quad \text{loss}_p(\theta) = -\sum_{n=1}^{N}\tilde{f}_\theta \log(s_{n+1}|s_n, a_n)$$

$$\text{Gaussian NN}\ \tilde{f} = \Pr(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \sum\nolimits_\theta(s_t, a_t))$$

这里每一个模型是一个高斯过程，基于当前 state、action 来预测下一步的 state，通过当前 state、action 来得到高斯分布的中心值和方差值，然后通过模型采样接下来的状态。每次采样时，通过 Ensemble 中的 N 个高斯里面采集一个高斯出来，做接下来的推演。通过这样的方式，可以 capture 环境模型中的两部分的 uncertainty。

一是认知不确定性（epistemic uncertainty），在某些状态情况下，由于数据缺乏的原因，会导致模型的推演不稳定，方差较大，因此需要通过 Ensemble 的方式来降低模型的不确定性；

二是固有的随机性（aleatoric uncertainty），在每一个模型中，在预测下一步状态时，会有一个自带的随机性，因此通过高斯建模的方式直接将这种随机性刻画出来。

最终，通过 ensemble 的高斯过程模型，首先采样出模型，得到一个状态分布，然后采样出接下来的状态，不断往前（forward）推演。通过这一过程，再结合使用 MPC 算法，从而可以得到有效的行动策略。

四、MBPO 算法理论分析与实现方法

另一方面，建立的环境模型与真实环境之间一定有偏差（error），这会导致从虚拟的环境模型中训练得到的策略，在虚拟环境中交互得到的 Value 值与在真实环境中交互得到的 Value 值有所差距，称为 value discrepancy bound（以下简称 bound）。人们将相应的差值定义为如下形式，左侧$\eta[\pi]$为真实环境中的 Value 值，右侧$\hat{\eta}[\pi]$为环境模型中的 Value 值及其相应 bound 差值。ε_π是策略自身所导致的偏移（policy shift），这是由于采集数据

时的策略为π_D，而当前采用策略为π，策略的不同会导致与环境交互产生的数据分布不同。ε_m为环境模型与真实环境之间的误差。

$$\eta[\pi] \geq \hat{\eta}[\pi] - \underbrace{\left[\frac{2\gamma\, r_{max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4 r_{max}\epsilon_\pi}{(1-\gamma)}\right]}_{C(\epsilon_m, \epsilon_\pi)}$$

$$\epsilon_\pi = \max_s D_{TV}(\pi || \pi_D)$$
$$\epsilon_m = \max_t \mathbb{E}_{s\sim\pi_{D,t}}[D_{TV}(p(s', r|s, a)\big|\big|p_\theta(s', r|s, a))]$$

基于以上分析，伯克利研究员提出一种新的基于模型的强化学习算法 MBPO (Model-based Policy Optimization)。如下所示，灰色模块所在的主干为智能体与真实环境交互时的一条轨迹。中间采样出来一个 state 后，由该 state 所在的 branch point 开始，使用模型进行一个 k-steps 的分支采样，如下面的蓝色模块所在的分支所示，并将采样出来的数据放到一个正常的 model-free 的强化学习算法中进行训练，从而提高数据采样效率，并降低模型与真实环境之间的 bound。

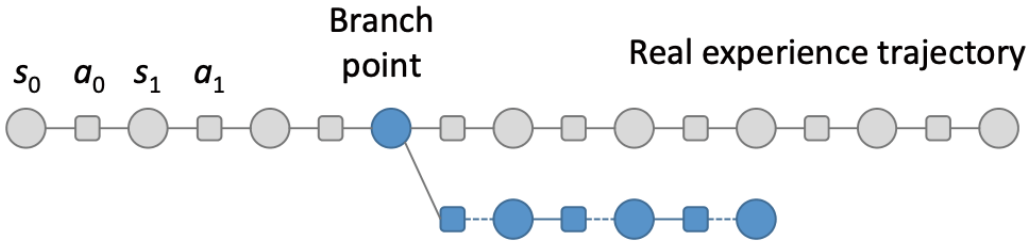


图 9：动态模型环境下的 k-steps 分支采样

在这个过程中，分叉后的推演步数 k 的取值起着重要的作用，人们推导了一个 bound 的计算方式:

$$\eta[\pi] \geq \eta^{branch}[\pi] - 2r_{max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k+1}}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_m + 2\epsilon_\pi)\right]$$

左侧$\eta[\pi]$为真实环境中的 Value 值，右侧$\hat{\eta}[\pi]$为环境模型中的 Value 值及其相应 bound 差值，根据上式计算，只有当 k=0 时 bound 最小，对应着不使用模型进行推演的情形。为了解决这一问题，人们重新查验推导过程，发现其中一步的ε_m中使用了基于之前时刻的 policy 采样得到的数据分布，此时如果换成使用当前时刻的 policy 采样得到的数据分布，如下所示:

$$\hat{\epsilon}_{m'}(\epsilon_\pi) \approx \epsilon_m + \epsilon_\pi \frac{d\epsilon_{m'}}{d\epsilon_\pi}$$

$$\epsilon_{m'} = \max_t \mathbb{E}_{s\sim\pi_t}[D_{TV}(p(s',r|s,a)||p_0(s',r|s,a))]$$

则可以得到一个非常有意义的结果:

$$\eta[\pi] \geq \eta^{branch}[\pi] - 2r_{max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k+1}}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_{m'})\right]$$

在新的公式中，通过对 bound 值的计算会发现，当 $\frac{d\epsilon_{m'}}{d\epsilon_\pi}$足够小，也就是当基于环境模型与真实环境之间的误差相比策略自身所导致的偏移（shift）足够小时，则 k>0 的时候会得到 bound 值最小的情形。

以 Mujoco 等环境为例进行分析会发现，$\frac{d\epsilon_{m'}}{d\epsilon_\pi}$的取值通常在 10^{-4}~10^{-2} 之间，能够满足相应的要求，从而为环境模型的使用提供了理论上的保证。

图 10：$d\epsilon_{m'}/d\epsilon_\pi$的经验分析

这一分析结果为人们处理为什么使用模型、如何使用模型的问题提供了思路。在解决实际问题的时候，可以去寻找一个合适的 k 值，使得这个 k 值能够保证 bound 达到最小。

如下是 MBPO 算法的主要流程，通过寻找基于当前模型的最好的 branch rollout 的步数 k，使得最后获得最佳的训练策略。这一流程基于黑盒模型，最后通过 model–free 的方法来对 branch rollout 的数据进行训练，比如 soft AC 等。

Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

Initialize policy π_ϕ, predictive model p_θ, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$
for N epochs **do**
  Train model p_θ on $\mathcal{D}_{\text{env}}$ via maximum likelihood
  for E steps **do**
    Take action in environment according to π_ϕ; add to $\mathcal{D}_{\text{env}}$
    for M model rollouts **do**
      Sample s_t uniformly from $\mathcal{D}_{\text{env}}$
      Perform k-step model rollout starting from s_t using policy π_ϕ; add to $\mathcal{D}_{\text{model}}$
    for G gradient updates **do**
      Update policy parameters on model data: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$

图 11：MBPO 算法的主要流程

下图是 MBPO 算法在不同环境下与其它算法的对比实验，通过 MBPO 的对比实验可以看出，MBPO 算法在训练效果上比较明显的领先于其它算法。

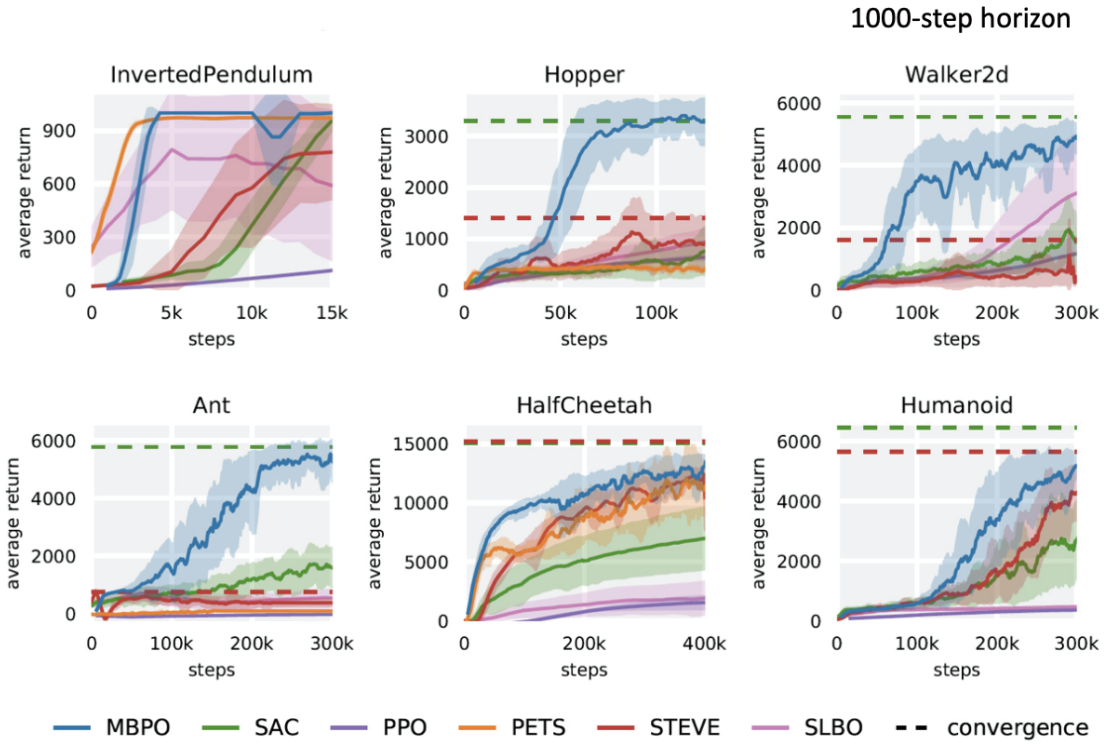


图 12：MBPO 算法的实验结果

五、最新的 BMPO 算法理论分析与实现方法

人们使用环境模型进行前向（toward）推演的时候，总是基于当前的 state 去逐步推演后续的 state，也就是当前时刻之后的状态。相应的，人们也可以建立环境模型，基于当前的 state 去逐步推演之前出现的 state，也就是当前时刻之前的状态。通过建立环境模型去推演到达当前时刻 state 之前，智能体所经历的轨迹，这样的模型被称为反向模型（backward model）。基于此，张伟楠等人在 ICML 2020 提出了一个新的算法：BMPO（Bidirectional Model-based Policy Optimization）。

推演过程中如果使用前向模型（forward model），从当前 state 向当前时刻后续进行推演，由于每一步推演都存在复合误差（compounding error），随着推演的不断进行，误差不断积累，模型推演出的结果与真实环境实际结果之间的差距将会逐渐增大，直至彻底失去作用。进行反向推演的意义在于，当使用双向模型（bidirectional model）同时进行前向与反向推演时，推演出的轨迹的总长度与使用单向模型一样的时候，前向模型与后向模型各自所需推演的步数均少于完全使用单向模型的情形，复合误差的积累要显著小于完全使用单向模型的情形。

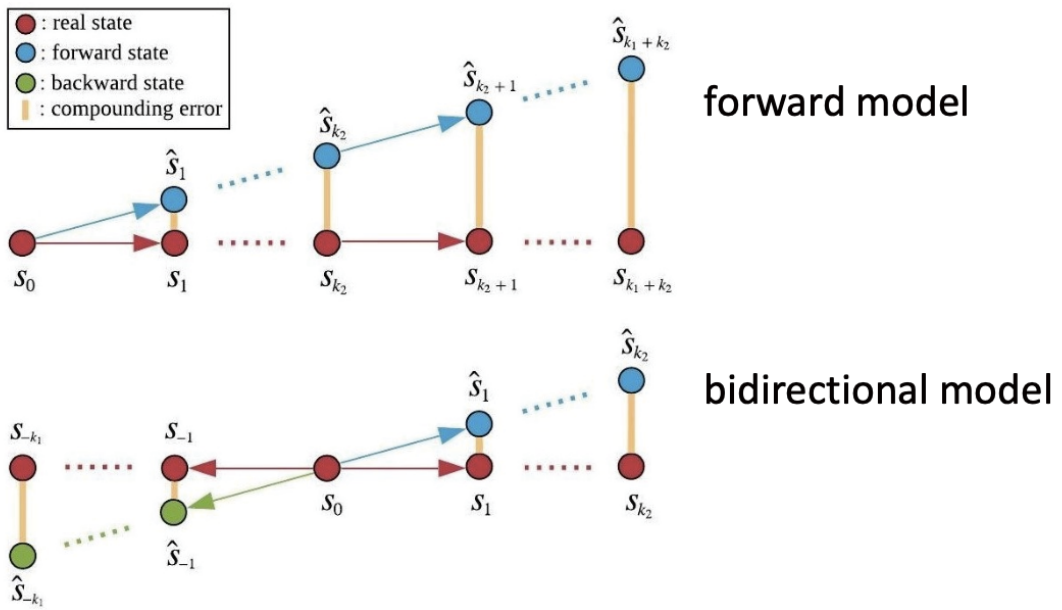


图 13：前向模型和后向模型

他们使用 PETS 方法，利用神经网络结合高斯过程的方式建模，建立反向推演的策略（backward policy），根据当前所在的状态，通过 MLE（maximum likelihood estimation）的方式学习之前 state 所采用的 action：

$$\mathcal{L}_{MLE}(\phi') = -\sum_{t=0}^{N} \log \tilde{\pi}_{\phi'}(a_t|s_{t+1})$$

也可以通过 GAN 的方式，来确定之前 state 下最可能采用的 action：

$$\min_{\tilde{\pi}} \max_{D} V(D,\tilde{\pi}) = \mathbb{E}_{(a,s')\sim\pi}[\log D(a,s')] + \mathbb{E}_{s'\sim\pi}[\log(1 - D(\tilde{\pi}(\cdot\,|s'),s'))]$$

通过这两种方式，便可以学习到反向推演的环境模型。

在 BMPO 的采样环节做 MBPO 的 branch rollout 时，根据玻尔兹曼分布来对选取具有更高价值函数的状态，以此来保证轨迹中有一个状态具有比较高的价值函数，从而可以通过双向模型来学习如何走到当前价值函数比较高的状态，以及之后选取动作的行动策略，从而最终获得最好的行动策略。

$$p(s) \propto e^{\beta V(s)}$$

另一方面，我们希望采样到高价值的状态。在策略与真实环境中交互过程中，结合当前模型，使用 MPC 的方式来采样获得带来更好价值函数的动作。

BMPO 的整个流程如上所述，首先通过 MPC 的方法来进行与环境交互的高价值数据的收集。通过收集到的数据建立模型，根据模型采样价值函数较高的状态作为双向 rollout 的初始状态。通过双向模型来进行双向推演，在获得的轨迹上使用 model-free 的方法（soft actor-critic）进行策略的训练。

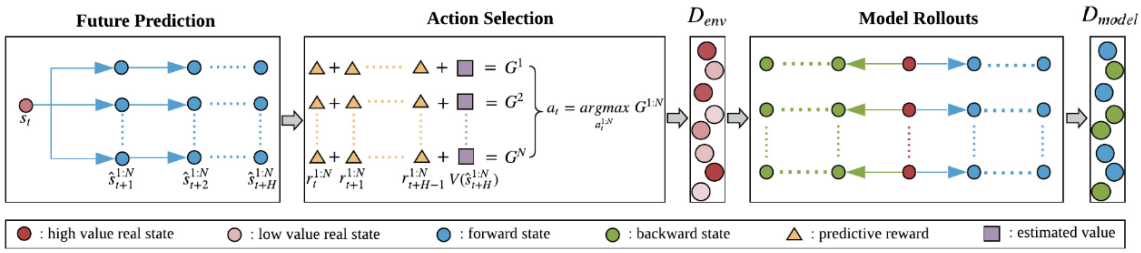
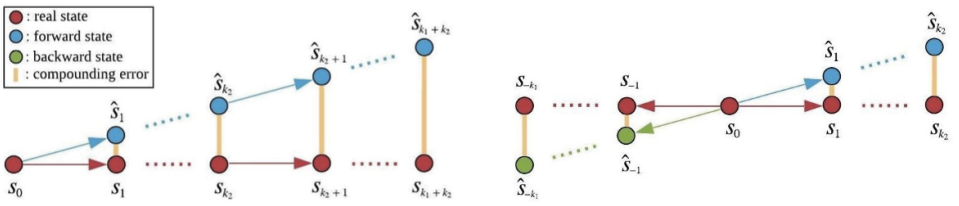


图 14：BMPO 的总体算法

张伟楠等人对 BMPO 与 MBPO 的结果进行了理论分析。当使用前向推演的单向模型（forward model）时，累积的复合误差为 k_1+k_2，而使用双向模型时，累积的复合误差为 $\max(k_1,k_2)$。由于 $\max(k_1,k_2)$ 一定小于 k_1+k_2，因此双向模型带来的复合误差会更小，训练效果也会更好。



Bidirectional model:

$$\left|\eta[\pi]-\eta^{\text{branch}}[\pi]\right| \leq 2r_{\max}\left[\frac{\gamma^{k_1+k_2+1}\epsilon_\pi}{(1-\gamma)^2}+\frac{\gamma^{k_1+k_2}\epsilon_\pi}{(1-\gamma)}+\frac{\max(k_1,k_2)\epsilon_m}{1-\gamma}\right].$$

Forward model:

$$\left|\eta[\pi]-\eta^{\text{branch}}[\pi]\right| \leq 2r_{\max}\left[\frac{\gamma^{k_1+k_2+1}\epsilon_\pi}{(1-\gamma)^2}+\frac{\gamma^{k_1+k_2}\epsilon_\pi}{(1-\gamma)}+\frac{(k_1+k_2)\epsilon_m}{1-\gamma}\right].$$

图 15：理论分析

最终基于 Mujoco 环境的对比实验结果证明了他们的分析：提出的 BMPO 算法能够比之前人们提出的 MBPO 算法具有更好的数据采样效率以及收敛效果，

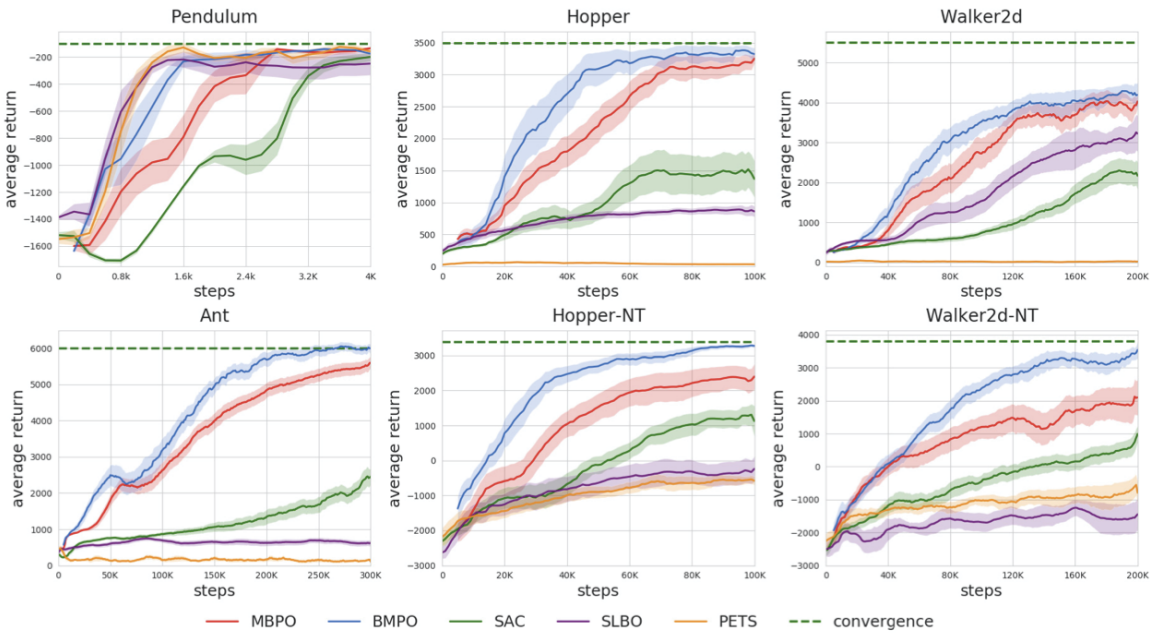
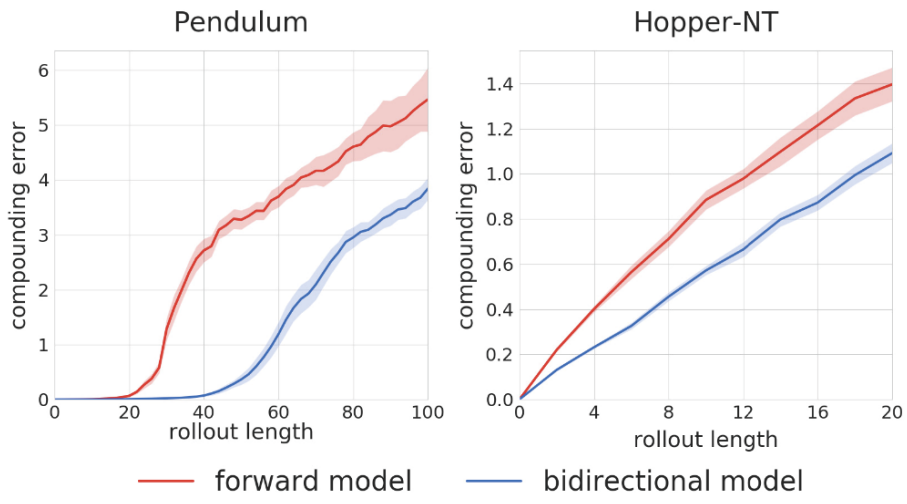


图 16：SOTA 比较

他们还对 BMPO 与 MBPO 的复合误差进行了量化的对比计算，结果发现在 rollout length 相同时，双向模型具有更小的复合误差。



$$\text{Error}_{\text{for}} = \frac{1}{2h}\sum_{i=1}^{2h}\|\hat{s}_i - s_i\|_2^2$$

$$\text{Error}_{\text{bi}} = \frac{1}{2h}\sum_{i=1}^{h}\left(\|\hat{s}_{h+i} - s_{h+i}\|_2^2 + \|\hat{s}_{h-i} - s_{h-i}\|_2^2\right)$$

图 17：复合误差

六、总结

MBRL 将会使接下来几年强化学习领域的研究热点，对于真实环境下的问题，我们都需要建立虚拟的环境来进行策略的训练与测试，之后才能应用于真实环境。MBRL 主要分为基于黑盒模型的 MBRL 以及基于白盒模型的 MBRL 两类，这项研究着重分析了基于黑盒模型的 MBRL。基于白盒模型的 MBRL 近期也出现了许多的成果，比如 ICLR 2020 的 MAAC 等算法。

对于 MBRL 未来的发展趋势，有如下几点:

一是当前研究中所使用的环境比较简单，比如 Mujoco 环境等。对于复杂的现实环境中的问题，比如研究电商中用户购买行为时，所面对的挑战难度非常大，因为这样的问题中具有非常多的不确定性，环境的 state 本身是离散的。人们所采用的高斯过程更适合连续的机械问题的建模，进行离散问题的建模时，一定会有偏差，这时偏差带来的累计误差就会非常大，面对复杂问题时，如何实现高质量的环境建模将是未来研究的热点。

二是如何将理论中推导出的 bound 使用到具体的算法中，MBPO 是一个非常好的实践，而未来会有更多的发展前景。

最后一点，MBRL 可以应用于多智能体强化学习（Multi-Agent RL，MARL）中，例如当前滴滴公司使用 MARL 处理车辆派单问题时，也会建立虚拟的环境模型来进行训练和测试，然后再在现实环境中进行部署。