开悟比赛-网络延迟 460 队技术整理与分享

高辰潇 南京大学 人工智能学院

韩馥光 南京大学 人工智能学院

俞睿 南京大学 人工智能学院

薛瑞奇 南京大学 人工智能学院

吴智超 南京大学 人工智能学院

指导老师:章宗长 赵一铮

一、 简介

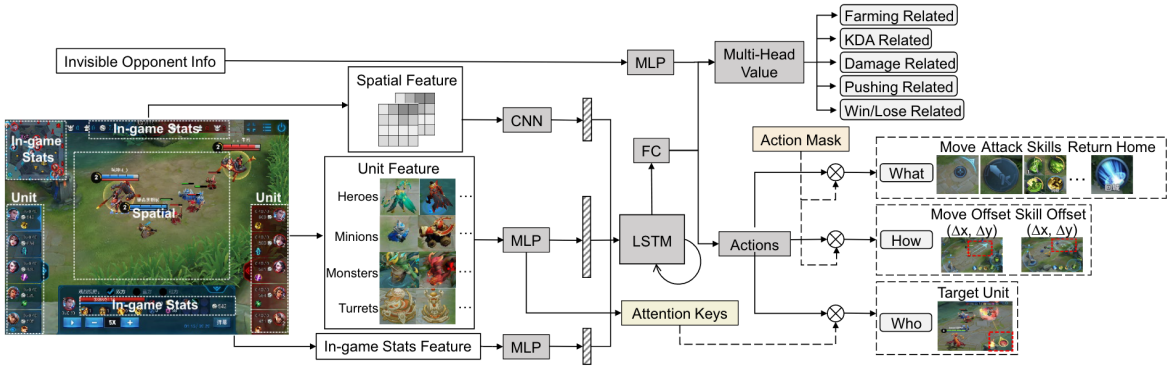
在 2022 年 9 月-2023 年 4 月举办的腾讯第三届开悟 MOBA 多智能体强化学习大赛中，我们队伍（网络延迟 460）在初赛中有幸获得了第一名的成绩。这得益于实验室老师与腾讯官方的大力支持， 也得益于我们队伍本身的技术探索与积累。本文首先会简单叙述比赛的基本情况，随后从各关键模块出发，简要介绍本队伍在开悟比赛中的探索历程与心得体会。

二、 参赛概况

我们队伍的成员都来自于南京大学人工智能学院，但是来自于不同的年级。其中，我（高辰潇）和韩馥光为研究生一年级的同学，其余三位成员均为本科三年级。大概在开学时，指导老师赵老师受腾讯的邀请在年级群中发布了有关 MOBA 多智能体强化学习大赛的资讯，由于开悟比赛与我个人的研究背景相契合，因此我就向赵一铮老师发送邮件报了名。同期其他几位同学也都报名参与了这次的比赛，他们有的同样具有强化学习的研究背景，希望能够在真实的应用场景下探索强化学习的应用；有的在王者荣耀这款游戏上技术娴熟，带着对比赛的热情和兴趣参与了这次比赛。在本次比赛期间，我们每周会固定地抽出时间线下交流、反馈、迭代算法，同时在线上保持密切交流，平均每人每周时间投入在 2.5 * 7 小时左右。

三、网络设计

我们神经网络的架构设计与提供的 baseline 高度一致，仅在部分细节上进行了修改。我们以论文[1]中的网络结构为例：



在这篇文章的结构的基础上，我们的网络结构有如下要点

1. 没有 spatial feature，所有的 feature 都是向量化表示的数据，因此不需要上图中所示的卷积网络。
2. 实现了 Multi-Head Value。我们发现多头值网络的设计能够有效提升 MOBA 这种奖赏定义比较复杂的场景下的智能体性能。同时还有一些额外的好处，比如发现 reward 定义存在问题、需要调整时，值网络中只有少量的输出头需要重新拟合，可以大大提高恢复训练时的稳定性。
3. 在各个动作分支上，我们加深了网络结构。Baseline 代码中的各个动作分支的网络层仅有一层网络层，我们认为这种情况下缺乏非线性变换的能力，因此我们为每一个动作分支额外添加了一个大小为 128 的隐藏层。实验发现这一改动对性能有显著的影响。
4. 为 value network 额外添加不可见的对手信息。尽管 MOBA 是一个 POMDP 场景，策略网络在执行时无法获取对方 agent 的完整状态，但是在训练时我们完全可以将这个不可见的状态提供给值网络以获得更为准确的值估计。
5. 为 action 和 value network 提供当前所控制英雄的身份信息。这次的比赛要求使用同一个网络模型控制不同的英雄，对网络的泛化能力的要求比较高。我们在 robust 和 adaptive 这两种 principle 中选择了后者，也就是通过向网络提供当前控制的英雄信息，以期网络能够自适应地采取更有针对性的决策。然而，由于框架代码的限制我们无法直接向网络提供英雄的身份，因此我们采用了一种间接的方式——利用当前英雄的 feature 额外训练一个身份分类器，预测当前英雄的身份，并将身份以 onehot 编码拼接在 LSTM 的输出之后作为后续网络层的输入。通过观察 loss 我们可以发现这么做是有效的，分类器的预测非常准确，我们推测是因为英雄的 feature 中存在一些能够指示英雄身份的量（例如最大 HP，移动速度等）所致。

四、奖励体系

我们对奖励体系的设计可以被归结为两点：

1. 局内衰减。局内衰减指的是一个 episode（一局游戏）之内 reward 随着时间的衰减。我们

希望在游戏的前期智能体能够更加关注稠密奖赏（比如经济、经验等），在游戏的后期能够更加关注一些稀疏但与游戏胜利直接相关的奖励（比如游戏胜利、KDA、推塔）。因此，我们设计了一种奖赏在局内衰减的机制，具体而言，使用了线性衰减/增益：

$$r_t = \text{clip}\left(\frac{r_{final} - r_{init}}{4000} * frame_no + r_{init}, r_{init}, r_{final}\right)$$

也就是基于 frame no 来线性衰减/增益奖赏，并将其 clip 到一定区间内。我们对不同奖赏的设置如下：

- 线性衰减：eprate(0.3-0.2), exp (0.008-0.003) money (0.008-0.003) last_hit(0.5-0.2)
- 线性增益：kill(0-0.4), hp(1-1.5), tower hp(5-7.5)

2. 局间衰减。局间衰减指的是 episode 之间、训练过程中的奖赏变动。在训练的前期，如果不给稠密奖赏设置更高的系数，智能体可能会因为奖赏信号过于稀疏而无法学习到好的行为模式。因此我们希望通过局间衰减，先让智能体学会对线和发育，然后在这个基础上学会推塔和击杀。

   局间衰减是通过暂停实验、保存模型、调整系数、重启实验来实现的，具体而言，我们在前期使用较高的经济和经验奖励，到后期略微调高击杀和胜负奖励的系数。

五、特征与规则

我们在初赛阶段没有进行任何规则后处理，因为当时觉得这可能会和 Policy Gradient 本身的优化过程相冲突。在特征方面我们也没有提取任何额外的特征。

六、 强化学习算法

在强化学习算法的选择上我们依然使用了 PPO 算法对策略进行优化。在 PPO 的基础上，我们参考[1]加入了 dual clip 和 value clip。

1. Dual clip。在原本 PPO 的基础上额外对 policy 进行一次 clip，从而避免 advantage 的下溢对策略优化的负面影响。

$$\mathcal{L}^{policy}(\theta) = \hat{\mathbb{E}}_t\left[\max\left(\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right), c\hat{A}_t\right)\right],$$

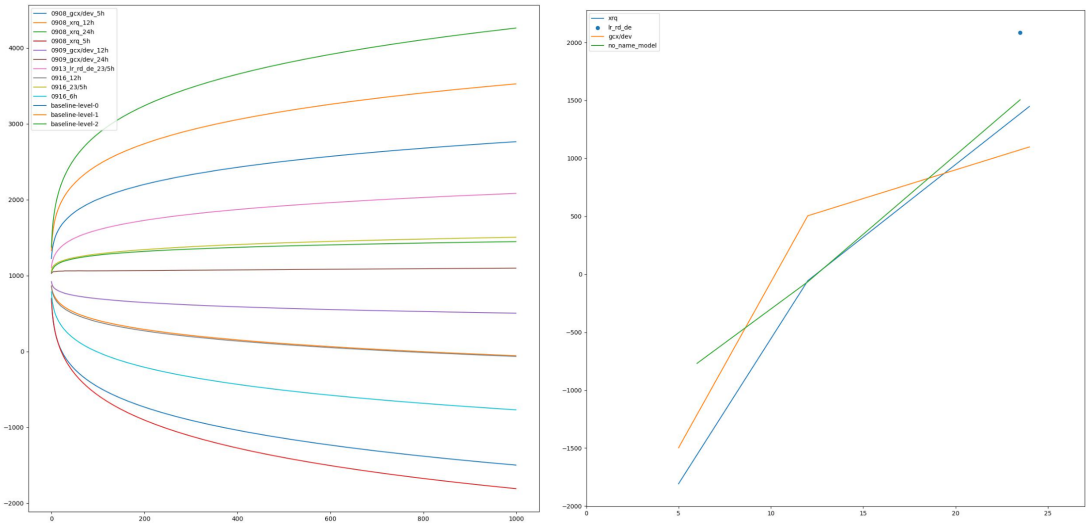
2. Value Clip。我们对值函数也进行了一次 clip，防止单次过于激进的值函数更新的负面影响。

$$\mathcal{L}^{value}(\psi) = E_t\left[\left(R_t - \left(V_{old} + \text{clip}(V - V_{old}, -\text{clip\_range}, \text{clip\_range})\right)\right)^2\right]$$

七、 系统工程架构

在系统工程架构上我们进行了比较多的探索与调整。

1. 对数据的陈旧度进行监控并进行轨迹截断。我们在 actor 端记录数据的生产时间，在 learner 端记录数据的消费时间，两者相减即可得到数据的陈旧程度。论文[2]表明，数据陈旧度越高，该数据在被生产时所使用的模型版本与当前版本相差便越大，会显著降低收敛速度和收敛性能。我们最开始观察到数据陈旧度在 90 秒左右，比较高，因此实现了轨迹截断机制。轨迹截断会定期暂停当前 episode 的采样、发送样本并加载新的模型进行采样，相比原来直到完整的对局结束才会发送样本的做法有效降低了数据的陈旧程度。这么做也有一定的缺陷，也就是截断采样并对最后一个时间步的状态进行估值会引入偏差，但是在实验中我们发现这个影响并不显著，遂忽略。
2. 自适应协调生产进程和消费进程的速率。[2]中同样发现，样本重用的次数也会影响算法的性能和收敛速度，较为理想的样本使用次数应该在 1 左右。由于 GPU 端消费数据的速度远高于 CPU 端生产数据的速度，我们需要对 GPU 进行一定的延迟。为了保持期望意义下样本使用次数为 1，我们通过监控 CPU 和 GPU 两侧的生产/消费速度，实现了一种自适应调整的方法。
3. 使用 ELO 对战机制对模型进行评价。我们参考现在的游戏匹配机制实现了 ELO 对战系统，对模型池中的候选模型进行自动对战与评分。具体而言，我们通过爬虫向开悟平台发起请求，自动选择模型池中的模型进行对战，并在对战完成后查询对战结果保存到本地。当所有的对战完成后，我们参考 ELO 机制对模型池中的所有模型进行评分，筛选出综合性能最优的模型作为下一次训练和迭代的基础。下图是我们所开发的 ELO 对战系统给出的分析图示。



八、 模型迭代过程

在模型的每一步迭代中，我们会进行如下操作

1. 观察比较经典的对战录像，并由同学分析对战数据。
2. 使用 ELO 系统筛选下一轮迭代的模型
3. 退火学习率、熵系数。这里使用的是退火，也就是总体上对学习率、熵系数进行衰减，但定期将学习率和熵系数调整为较大的值。这样做是为了能够通过类似模拟退火的想法跳出局部最优，对潜在的最优策略进行搜索。
4. 对 reward 系数进行局间衰减。
5. 重启实验。

九、 训练效果分析

我们在这一章节总结最终提交的模型的一些特点和技术指标。

在训练过程上，我们发现多头值网络架构、加深策略网络结构、对奖赏系数进行局内和局间衰减/增益和熵系数的退火带来了显著的性能提升，但是由于算力限制，我们无法隔离出各个模块进行消融实验以确定单个技术对算法渐进性能的影响。

在最终性能上，我们的模型能够比较稳定地战胜 baseline-level-4，并且对战不同风格的对手模型都能够稳定发挥，我认为这是我们引入 ELO 评估机制所带来的收益。

在多英雄泛化问题的处理上，我们本次并没有增加额外的结构或设计不同的训练流程，仅仅在策略网络的输入上添加了英雄 ID 的 embedding 作为 context，希望网络能够做到适应。但由于策略网络本身建模层面的一些问题（例如，策略网络的输出头往往是动作空间上的高斯分布），这样的设计根本无法建模 multi-modal 的动作分布——当不同英雄的差异过大时，策略网络很难输出差异化的动作分布。这一现象在我个人的科研实践中也有观察到，同时上海交通大学许志钦老师的工作也在神经网络的 inductive bias 层面对此有过介绍。回到我们的模型，在训练的后期我们发现模型开始出现一种“田忌赛马”的倾向，也就是主动放弃弱势英雄（马可波罗），而不断精进强势英雄（鲁班）的控制。很可惜在初赛阶段受限于训练资源，我们没有能够深入探讨；在复赛阶段我们在网络参数的共享和隔离上对解决这一问题进行了初步尝试。

十、 总结与展望

作为首次参加 MOBA 多智能体强化学习大赛的队伍，本次初赛阶段我们熟悉了任务环境、跑通了训练流程、进行了一定程度的算法设计并有幸获得了第一名的好成绩。很可惜受限于时间，我们还未能对以下问题进行探讨：

1. 多英雄泛化问题。多英雄泛化问题本周上对应于一个 multi-task 任务，我们希望能够在复赛和决赛阶段借助 multi-task 中的技术帮助模型在多个英雄甚至多种英雄的控制上取得良好表现；
2. 模型池与多风格对手对战。受限于框架代码的限制，我们发现很难实现 OpenAI five 中的多风格对手池。希望框架代码能够开放相关接口，进一步释放算法设计的空间。

以上便是我们对于本次初赛的总结。

参考文献

[1] Towards Playing Full MOBA Games with Deep Reinforcement Learning.

[2] Grandmaster level in StarCraft II using multi-agent reinforcement learning.