# 一，阅读比赛信息

找到了中文版（mineRL competition 相关信息），对照原版，进行了详细阅读

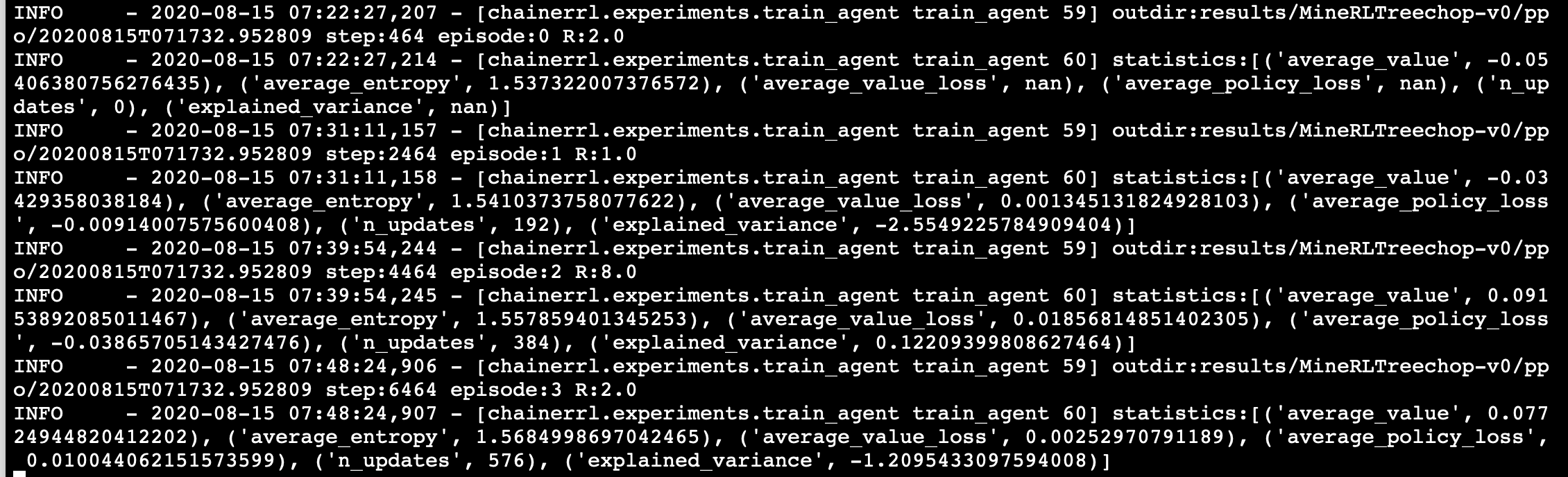
# 二，搭建环境

在服务器上；python环境配置正常，但java环境一直有问题

在本地：使用docker可以跑

# 三，跑baselines

使用docker，跑起来了，但是cuda不匹配用不了显卡，训练速度比较慢



由于服务器刚刚弄好，后续分析及结果持续将更新......

# 四，算法简述

## ppo：

PPO(Proximal Policy Optimization),汉语意思近端策略优化

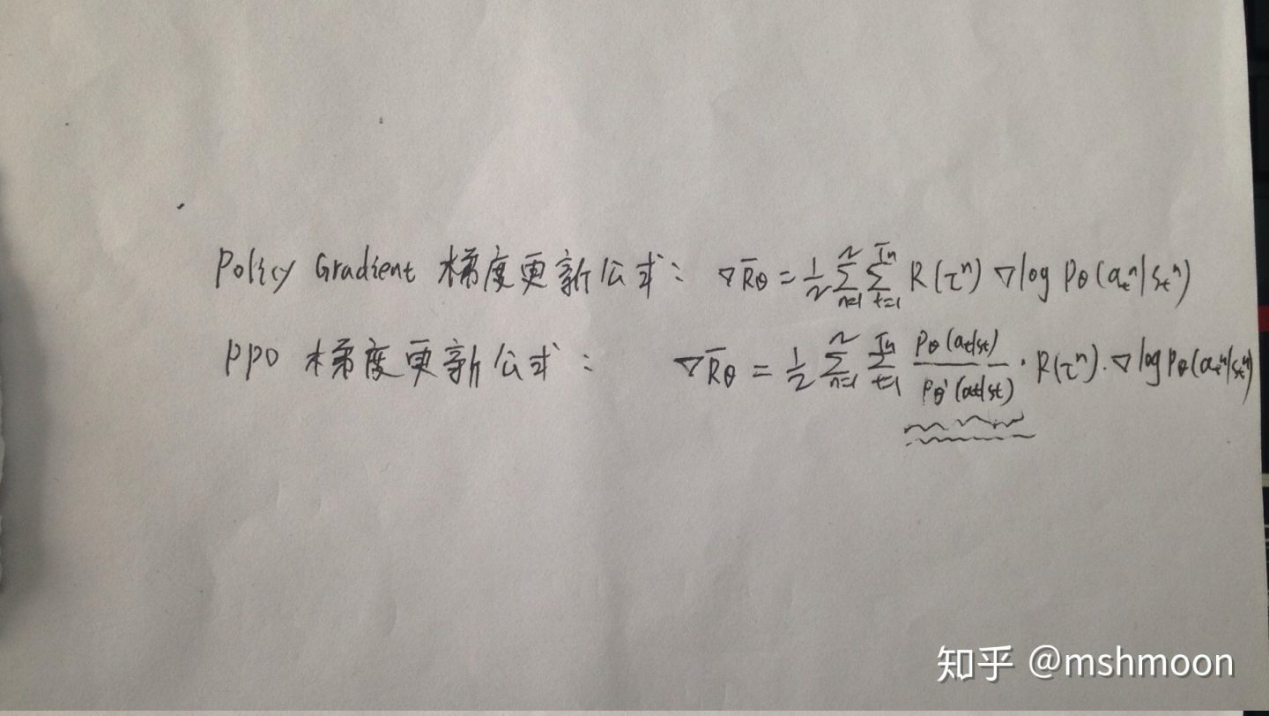
离线学习策略 PPO解决深度强化学习中样本只能使用一次的问题。

主要使用：裁剪的替代⽬标函数 Clipped Surrogate Objective

将剪裁参数线性衰减为零

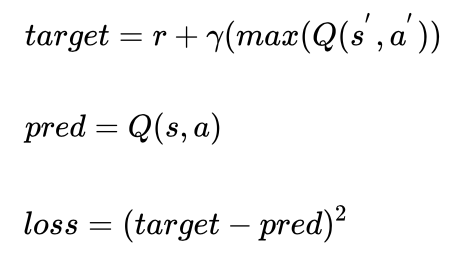
**以下摘自知乎：**

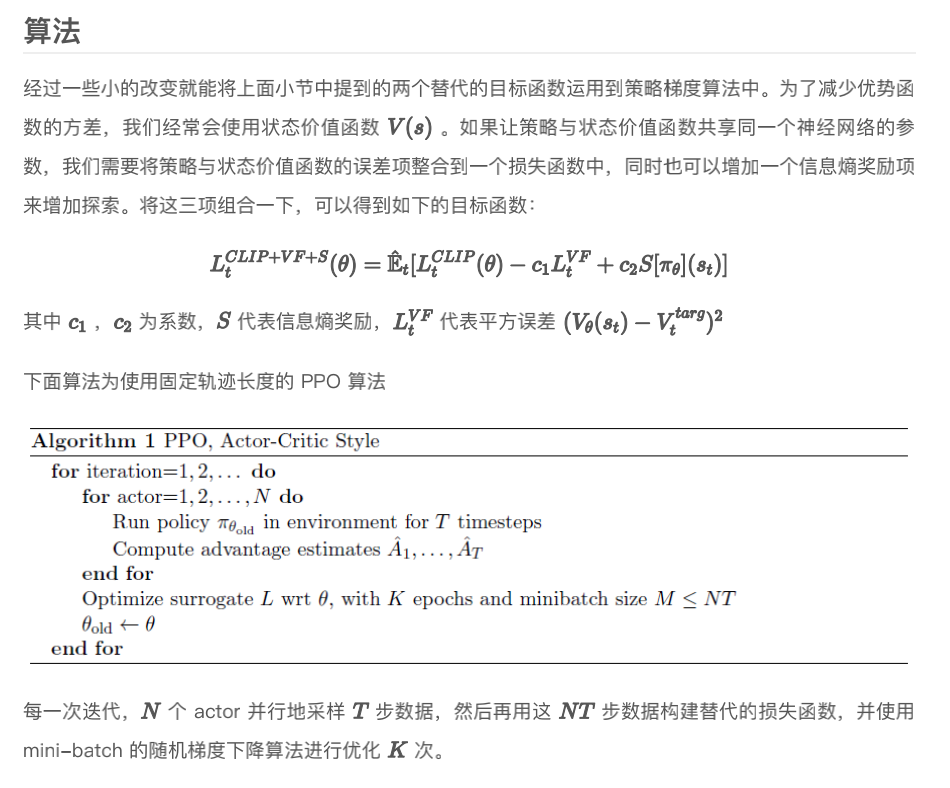
重要性采样要解决的问题是当我们想要从一个分布中采样，但是采样很困难，那我们可以从另一个容易采样的分布中采样，但是这2个分布不同，我们需要把采样到的样本做一个修正。把这个思想用到Policy Gradient中就是我们用策略pi去和环境互动，收集到一大堆数据，然后我们使用策略pi去学习这些数据，然后更新策略pi为pii，此时策略pi已经改变为pii，也就是说我们使用pii去和环境互动，很可能收集不到这样一堆数据，因此我们也就不能使用pii去学习这一堆数据。但是，我们还想使用pii去学习这一堆数据，那就需要修正这一堆数据



从公式中可以看出 Policy Gradient 梯度更新公式比 PPO 的梯度更新公式多乘了一项 [公式]，这就是修正项。 分子p是当前策略，分母 [公式] 是采样策略。也就是说，我用采样策略采集一大堆数据，用策略p去学习，当策略p发生改变时，只须用修正项修正即可。但是在实际应用中，采样策略也是要被更新的，一般每隔一个epoch更新一次采样策略。

当PPO与actor-critic框架结合时， agen的由两部分组成，一部分actor，负责与环境互动收集样本，另一部分是critic，负责评判actor的动作好坏。actor的更新即可使用PPO梯度更新公式，critic的loss的计算如下:





找到一篇博客，参考其中的算法：



## dqn\_family：

rainbow 算法整合了6种dqn算法：

简单机翻一下论文：

在代理方面，操作选择是由策略π给出的，该策略定义了对每个状态的操作的概率分布。 从时间t处遇到的状态ST，我们定义了折扣返回 作为代理收取的未来奖励的折扣总和，其中未来奖励k步骤的折扣是由折扣在此之前的乘积给出的， 。 代理的目标是通过找到一个好的政策来最大化预期的贴现回报。

该策略可以直接学习，也可以作为其他学习量的函数构造。 在基于价值的强化学习中，代理学习预期的disou的估计当遵循从给定状态开始的策略π时，vπ(S)=，或状态-动作对，qπ(s,a) ==[。 一种从新政策中得出新政策的共同方法 状态-动作价值函数是对动作价值贪婪地行动。 这对应于以概率（1)的最高值(贪婪行为）采取的行动，以及OT hher wise行为一致随机，概率。 这类策略被用来引入一种探索形式：根据其当前的估计，随机选择次优的动作在适当的时候，代理可以发现和纠正它的估计。 主要的限制是很难发现延伸到未来的其他行动方针；这促使了对更有针对性的探索形式的研究。

DQN是一个重要的里程碑，但该算法的几个局限性目前已知，并提出了许多扩展。 我们建议选择六个扩展，每个扩展已经解决限制和提高整体绩效。 为了使选择的大小易于管理，我们选择了一组扩展来解决不同的问题(例如，只是许多解决问题的探索之一 艾辛)。

主要有以下6个：

**Double Q-learning. Prioritized replay. Dueling networks.**

**Multi-step learning. Distributional RL. Noisy Nets.**

使用综合代理

