机器学习工程师纳米学位毕业项目

DeepRL 四轴飞行器控制器

李旭

2018 年 12 月 2 日

目　录

I 定义

1.1 项目概述

四轴飞行器控制为无人机领域的核心，根据前瞻产业研究院发布的《2017-2022 年中国无人机行业市场需求预测及投资战略规划分析报告》数据显示，娱乐/消费级无人机的高渗透率直接促进了市场扩张，工业级无人机应用领域也随之扩大，从农用植保、电力/石油管道巡检、到防恐救灾、警用安防等，潜在市场空间极大。

该报告预测 2018 年全球无人机市场销售收入将突破 80 亿美元，保守估计到 2025 年，行业市场规模将超过 700 亿美元。

对于无人机使用，发挥无人机价值最大化，很重要的内容是如何控制完成一整套飞行任务，比如让它学习起飞、悬停并稳定降落。这是 Reinforcement Learning 要解决的问题：让 agent 学习在一个环境中的如何行为动作(act)， 从而获得最大的奖励值总和(total reward)，这个奖励值一般与 agent 定义的任务目标关联。由于这套飞行任务又是连续性的，需要连续行为的策略学习，因此我们选择使用 DDPG 算法来完成。

1.2 问题陈述

四轴飞行器的姿态控制是非常复杂的，需要空气动力学、机械、电子硬件、软件等人才一起合作才可以完成的大工程。因此，在四轴飞行器控制器设计中，如何设计一个通过深度强化学习智能体，来控制四轴飞行器的飞行任务，包括起飞、悬停和着陆等一整套操控，是非常需要解决的问题。通过对一整套动作拆解，分开训练三个（子）智能体，让它学习起飞（至少抵达地面以上 10 个单元）、悬停（同样地，至少在地面以上 10 个单元的位置悬停）并缓慢降落。并分别对各阶段绘制阶段奖励，从而评价强化学习效果。

1.3 评价指标

本项目中，将基于行动者-评论家（Actor-Critic，AC）框架算法为基准模型，使用这一基准模型进行训练，得出其 reward 与最终行为效果，再使用 DDPG 算法训练，得出其 reward 与最终行为效果； 通过上述两个模型实现包含三个子智能体的一个完整的飞行系统。通过将 state，reward 和 done 值传递给智能体的 step() 方法，智能体将返回一个动作向量，该向量与你所定义的动作空间一致，前三个元素被解释为 x，y 和 z 方向上的三个力，其余 3 个元素分别表示要在这些轴周围施加的扭矩。得分 score 通过回报 reward 计算获得。因此可以通过定义不同阶段不同的奖励机制完成对各阶段基准模型和求解模型性能的评价与对比，以分析 DDPG 性能是否更优越。

2 分析

2.1 数据的探索

在本项目中，我们定义了一系列可供使用的其他参数：cube_size （四轴飞行器飞行环境立方块大小，为智能体观察环境，检查飞行器偏离立方块空间中心点大小），timestamp（你可以使用这个参数来检查是否超时，或是计算速度），pose（四轴飞行器的位置和方向，是一个包含七个元素的状态向量），angular_velocity, 和 linear_acceleration。通过将状态传递给智能体，来计算奖励值，并检查该阶段是否完成（比如智能体超过了时限，或是达到了特定高

度)。其中 reward 和 done 都基于智能体在过去做出的动作。通过将 state，reward 和 done 值传递给智能体的 step() 方法，智能体将返回一个动作向量，该向量与定义的动作空间一致，动作向量的前三个元素被解释为 x，y 和 z 方向上的三个力，其余 3 个元素分别表示要在这些轴周围施加的扭矩。

2.2 算法和技术

2.2.1 Actor-Critic 算法

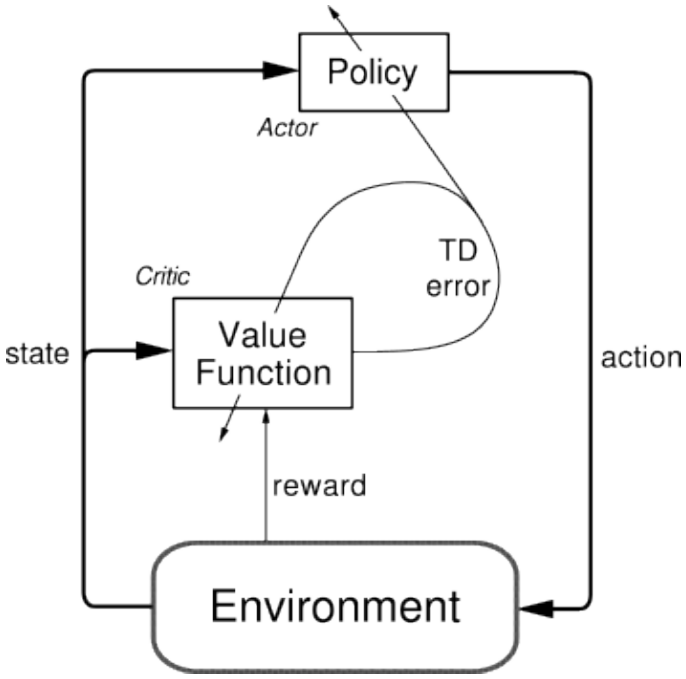
Actor Critic 为类似于 Policy Gradient 和 Q-Learning 等以值为基础的算法的组合。算法如下：

其中 Actor 类似于 Policy Gradient，以状态 s 为输入，神经网络输出动作 actions，并从在这些连续动作中按照一定的概率选取合适的动作 action。

Critic 类似于 Q-Learning 等以值为基础的算法，由于在 Actor 模块中选择了合适的动作 action，通过与环境交互可得到新的状态 s_, 奖励 r，将状态 s_作为神经网络的输入，得到 v_，而原来的状态 s 通过神经网络输出后得到 v。

通过公式得到状态之间的差 ，最后通过状态 s，动作 action，以及误差更新 Actor 网络的参数，实现单步更新。

将 s_ 状态赋予给 s 状态。



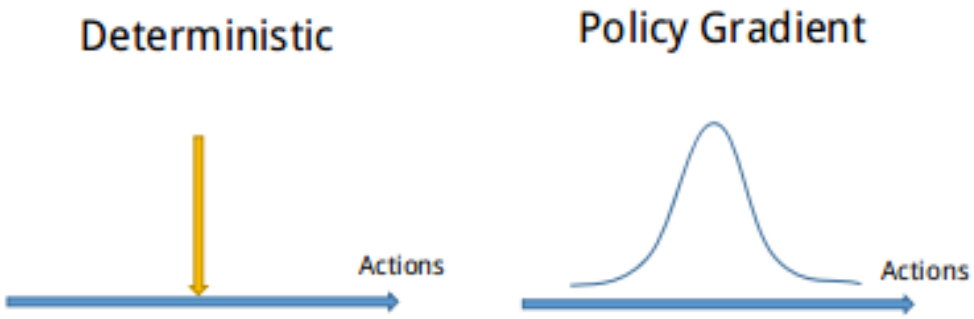
所以，Actor Critic 结合了 Policy Gradient（Actor）和 Function Approximation Critic）。Actor 基于概率选择行为，Critic 基于 Actor 的行为评判行为的得分，Actor 根据 Critic 的评分修改选择行为。

Actor Critic 方法的优势：可以进行单步更新，比传统的 Policy Gradient 要快。

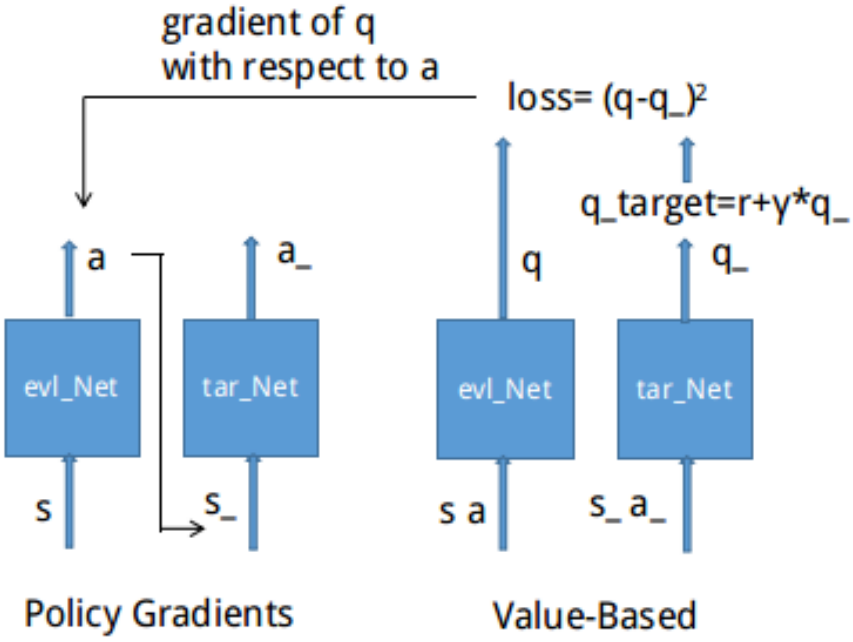
Actor Critic 方法的劣势：取决于 Critic 的价值判断，但是 Critic 难收敛，再加上 Actor 的更新，就更难收敛了。为了解决收敛的问题，有改进版的 Deep Deterministic Policy Gradient （DDPG），不仅仅融合了 DQN 的优势，同时解决了难收敛的问题。

2.2.2 DDPG 算法

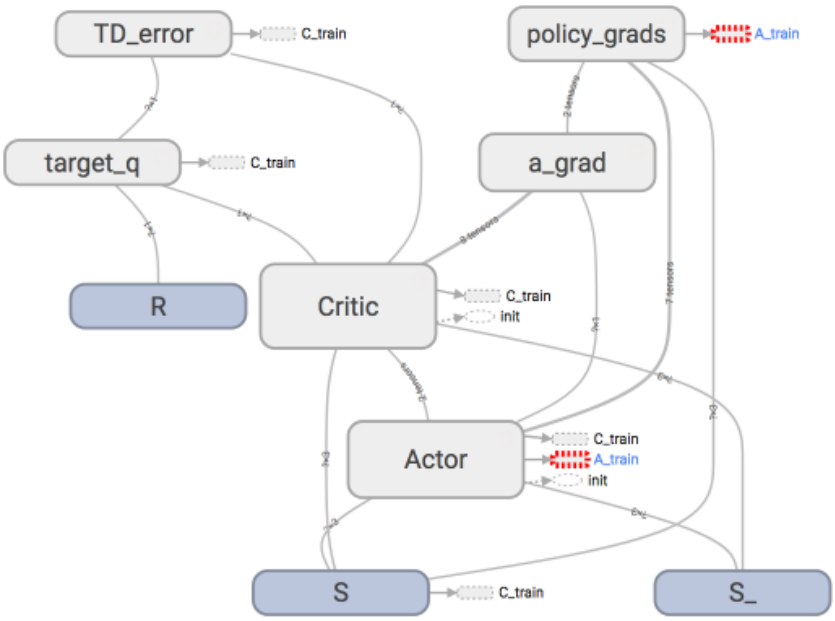
DDPG （Deep Deterministic Policy Gradient）可以分为 'DEEP' 和 'Deterministic Policy Gradient'，然后'Deterministic Policy Gradient' 又能被细分为 'Deterministic' 和 'Policy Gradient'。其中 DEEP 就是走向更深层次，使用一个记忆库和俩套结构相同，但是参数更新频率不同的神经网络有效促进学习。采用的神经网络类似于 DQN，但是 DDPG 的神经网络形式比 DQN 要复杂一点点。



接着是 Deterministic Policy Gradient，如上图所示，相比其他强化学习方法，Policy gradient 能被用来在连续动作上进行动作的筛选，而且筛选的时候是根据所学习到的动作分布随机进行筛选。而 Deterministic 改变了输出动作的过程，旨在连续动作上输出一个动作值。



DDPG 用到的神经网络有点类似于 Actor-Critic，也需要有基于策略 Policy 的神经网络 和 基于价值 Value 的神经网络，



但是为了体现 DQN 的思想，每种神经网络我们都需要再细分成俩个。Policy Gradient 这边有估计网络和现实网络，估计网络用来输出实时的动作，而现实网络则是用来更新价值网络系统的。再看看价值系统这边，我们也有现实网络

和估计网络，他们都在输出这个状态的价值，而输入端却有不同，状态现实网络这边会拿着当时 actor 施加的动作当做输入。在实际运用中，DDPG 的这种做法的确带来了更有效的学习过程。DDPG 的更新公式可从上面得出，关于 Actor 部分，他的参数更新会涉及到 Critic， 上面是关于 Actor 参数的更新，它的前半部分 grad[Q] 是从 Critic 来的，阐述了 Actor 的动作要怎么移动，才能获得更大的 Q。而后半部分 grad[u] 是从 Actor 来的，阐述了 Actor 要怎么样修改自身参数，才使得 Actor 有可能做这个动作，所以俩者阐述了 Actor 要朝着更有可能获取 Q 的方向修改动作参数。

2.2.3 技术

项目中将会使用 Keras， Keras 是一个高层神经网络 API，Keras 由纯 Python 编写而成并基于 Tensorflow、Theano 以及 CNTK 为后端，Keras 默认使用 TensorFlow 作为后端来进行张量操作。Keras 为支持快速实验而生，其优势如下：

- 简易和快速的原型设计（keras 具有高度模块化，极简，和可扩充特性）
- 支持 CNN 和 RNN，或二者的结合
- 无缝 CPU 和 GPU 切换

Keras 的设计原则：

用户友好：Keras 是为人类而不是天顶星人设计的 API。用户的使用体验始终是我们考虑的首要和中心内容。Keras 遵循减少认知困难的最佳实践：Keras 提供一致而简洁的 API， 能够极大减少一般应用下用户的工作量，同时，Keras 提供清晰和具有实践意义的 bug 反馈。

模块性：模型可理解为一个层的序列或数据的运算图，完全可配置的模块可以用最少的代价自由组合在一起。具体而言，网络层、损失函数、优化器、初始化策略、激活函数、正则化方法都是独立的模块，你可以使用它们来构建自己的模型。

易扩展性：添加新模块超级容易，只需要仿照现有的模块编写新的类或函数即可。创建新模块的便利性使得 Keras 更适合于先进的研究工作。

与 Python 协作：Keras 没有单独的模型配置文件类型（作为对比，caffe 有），模型由 python 代码描述，使其更紧凑和更易 debug，并提供了扩展的便利性。

2.3 基准模型

本项目使用 DDPG 算法，是对确定性策略梯度（Deterministic Policy Gradient， DPG）方法进行改造，提出的一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。DDPG 是异策略的方法，行为策略为随机策略，评估策略为确定性策略。随机策略可以探索和产生多样的行为数据，确定性策略利用这些数据进行策略的改善。通过对智能体的行为与状态的探索改善，基于策略 Policy 的神经网络 和基于价值 Value 的神经网络完成，目的是探索潜在的更优策略，在训练过程中，为 action 的决策机制引入随机噪声，将 action 的决策从确定性过程变为一个随机过程， 再从这个随机过程中采样得到 action，下达给环境执行.

3 方法

3.1 数据预处理

首先对环境空间数据进行设置，如 cube_size 为 3000，单位为 units，这样就可以分别对重心位置，目标位置，速度，行为空间，观察空间等进行设置，并且单位统一化。并且对速度及停留时间等进行初始设置。

3.2 执行过程

首先，通过将项目分为任务子目录 tasks（环境）及学习智能体子目录 agents，添加任务基类及定义 reset() 和 update() 方法，用于在运行子任务时从基类中继承，并在执行时覆盖基类中定义的方法，完成任务运行。对于 reset()方法，为重置/初始化变量，来为下一个阶段做准备。使用时并不需要自行调用它，它将被从外部调用。在每一个任务前，它都将被调用。因此在执行第一个任务 Takeoff 中，并没有任何可初始化的变量，但它必须为任务返回一个有效的_初始状态_，即元组对象。这些都是 ROS 信息类型，在任务开端，

可以向四轴飞行器传送这些动作（位置和方向）和速度（长度和角度）信息。在每个任务中，可选择同样的初始值，也可做出一些改变，比如在 `Takeoff` 中使四轴飞行器的高度随意下降一些等。当准备好将状态传递给智能体时，需要计算奖励值，并检查该阶段是否完成（比如智能体超过了时限，或是达到了特定高度）。reward 和 done 都基于智能体在过去做出的动作。

其次，对 update() 方法，来定义任务的动态并传达给智能体。它由一个 ROS 程序进行周期性调用（默认值为每秒 30 次），并与模拟中的当前数据一起，还有一系列可供使用的其他参数：timestamp（可以使用这个参数来检查是否超时，或是计算速度），pose（四轴飞行器的位置和方向），angular_velocity, 和 linear_acceleration。你不需要在每个任务中都使用这些参数，比如 Takeoff 只需要用到位置信息，而这是一个包含七个元素的状态向量。

通过将 state，reward 和 done 值传递给智能体的 step() 方法，智能体将返回一个动作向量，该向量与已定义的动作空间一致，如为 Box(6,)。在检查该动作向量不是空向量，并将它限制在空间限制之内后，需要将它转换为一条 ROS Wrench 信息。该动作向量的前三个元素被解释为 x，y 和 z 方向上的三个力，其余 3 个元素分别表示要在这些轴周围施加的扭矩。从 update() 方法返回 Wrench 对象或者 done 标志。 这将作为控制命令返回到模拟器，并且会影响四轴飞行器的姿态、方向和速度等。因此，在下一个时间步中调用 update() 方法时，我们将能够评估效果。在不同任务中，通过 DDPG 算法，设置不同任务的初始参数及奖励函数，完成智能体的训练及相应起飞，悬停，着陆任务。

3.3 完善

在任务运行过程中，不断调整 cube_size 大小，以适应起飞，悬停，着陆过程中，智能体的充分学习过程。设置 max_duration 时长，以满足在完成各子智能体及一整套智能体动作完成的时长保证及满足 400 次 Episode 的效率需求。在使用 DDPG 算法中，使用了 Keras CNN，用了 2 层隐藏层 全连接层 (Dense)，activation 为 'relu'， 最终一层 activation 为 sigmoid，参数 gamma 为 0.99， tau 为 0.001 以满足智能体完成强化学习的需要。

4 结果

4.1 模型的评价与验证

在本项目中，通过对智能体的起飞，悬停，着陆任务的定义，由于行动空间由连续变量组成，影响着四轴飞行器的姿态、方向和速度等。由于这套飞行任务又是连续性的，需要连续行为的策略学习，因此我们选择使用 DDPG 算法来学习训练一个智能体，让它顺利从地面起飞，并抵达特定高度完成任务。通过设置起飞参数，最终起飞模型能达到比较合理的结果。 在智能体分别完成起飞，悬停，着陆任务时，设置参数 max_duration 为 5 秒是一个比较合理的，当减少这个值时，智能体的一系列动作较少，学习时间太短，结果不理想； 当这个时长较长时，周期太长，影响学习性能。当智能体在完成起飞，悬停，着陆整套任务时，设置参数 max_duration 为 7 秒，能够充分较好的保障这一整套任务的可靠性学习，并有一个比较好的结果。通过设置折扣因子设置 0.99，以提高回报，促进智能体学习训练并对外来奖励进行考虑。因为 gamma 是考虑未来奖励的因子，如果 gamma 太小，比如 0.3，我发现终点处的正奖励不能够“扩散”到周围，智能体可能无法很好的学习并完成任务。为了能够充分地对外来奖励进行考虑，我们取值 gamma 为 0.99。基于对模型的分析及智能体学习过程及完成任务的需求考虑，我认为起飞，悬停，着陆及一整套任务模型是可靠、可信的。

4.2 合理性分析

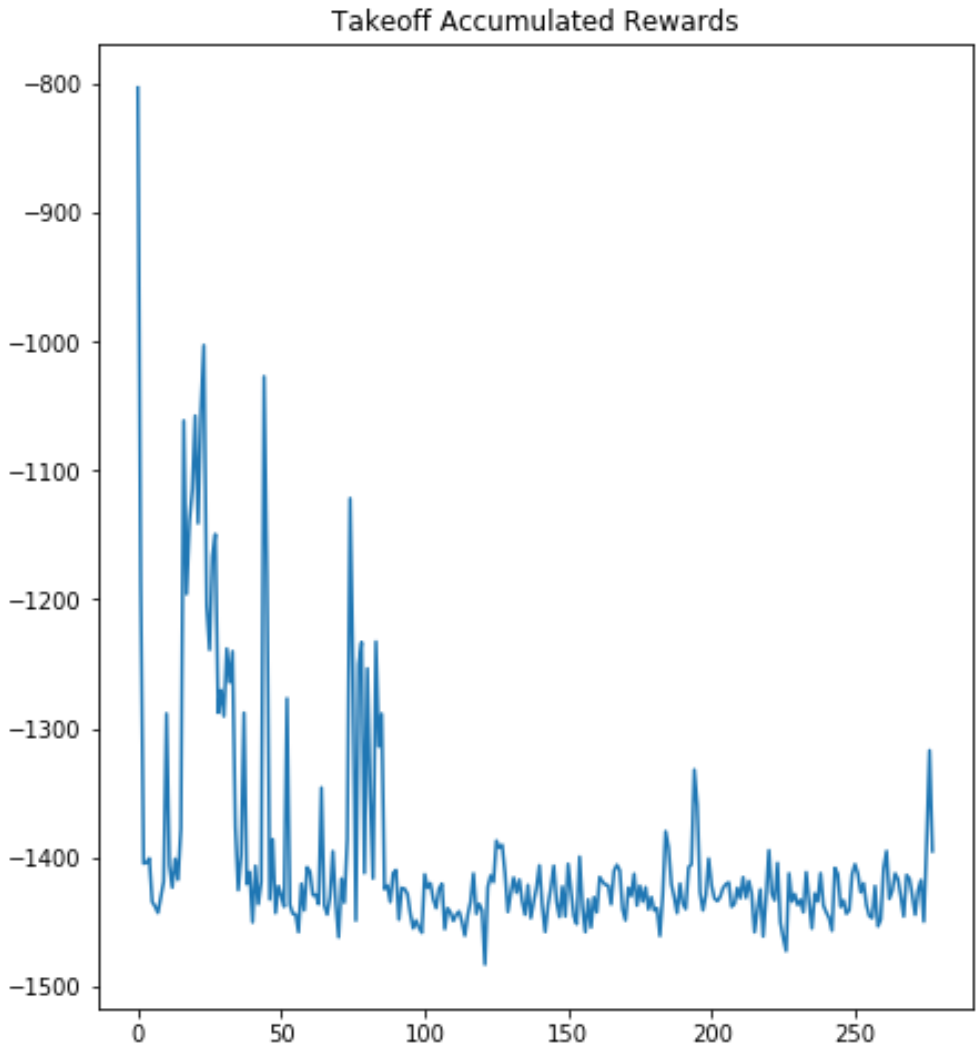
在本项目使用 DDPG 算法，是对确定性策略梯度（Deterministic Policy Gradient， DPG）方法进行改造的，是一种基于行动者-评论家（Actor-Critic，AC）框架的算法，该算法可用于解决连续动作空间上的 DRL 问题。DDPG 是异策略的方法，行为策略为随机策略，评估策略为确定性策略。随机策略可以探索和产生多样的行为数据，确定性策略利用这些数据进行策略的改善。通过对智能体的行为与状态的探索改善，基于策略 Policy 的神经网络 和 基于价值 Value 的神经网络完成，目的是探索潜在的更优策略，在训练过程中，为 action 的决策机制引入随机噪声，将 action 的决策从确定性过程变为一个随机

过程， 再从这个随机过程中采样得到 action，下达给环境执行。与 Actor-Critic 算法基准模型相比，表现更好些。通过 DDPG 算法运行任务所产出的回报与 episode 数据，我们发现起飞任务及着陆任务相对结果比较好， 能够很快收敛，效率较高。能够解决四轴飞行器控制起飞、悬停、着陆问题。

5 项目结论

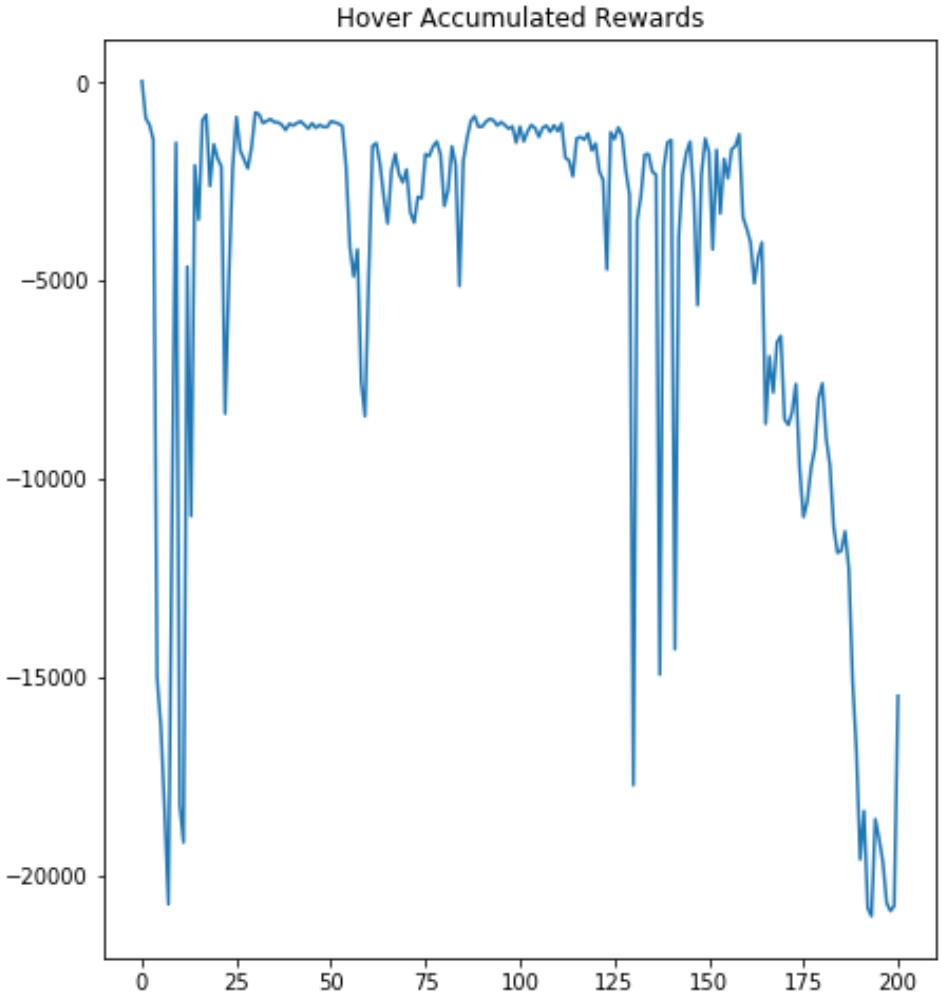
5.1 结果可视化

1, Takeoff 任务结果（X 轴 episode， Y 轴 Rewards）：



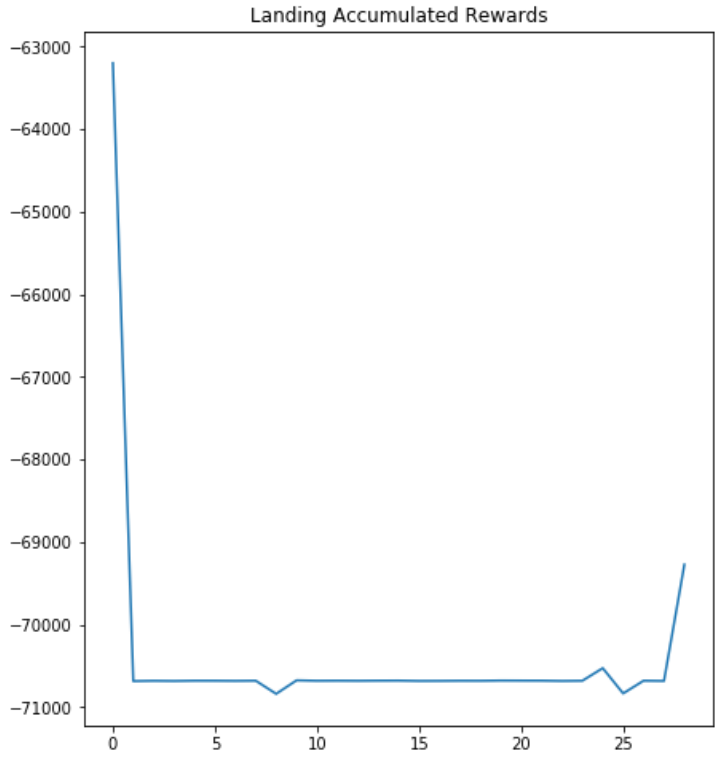
通过 Takeoff 的 episode 与 Rewards 数据结果能够看到，当 episode 在 80 后，Rewards 能够比较稳定的收敛，回报达到最高，性能比较稳定。

2, Hover 任务结果（X 轴 episode， Y 轴 Rewards）：



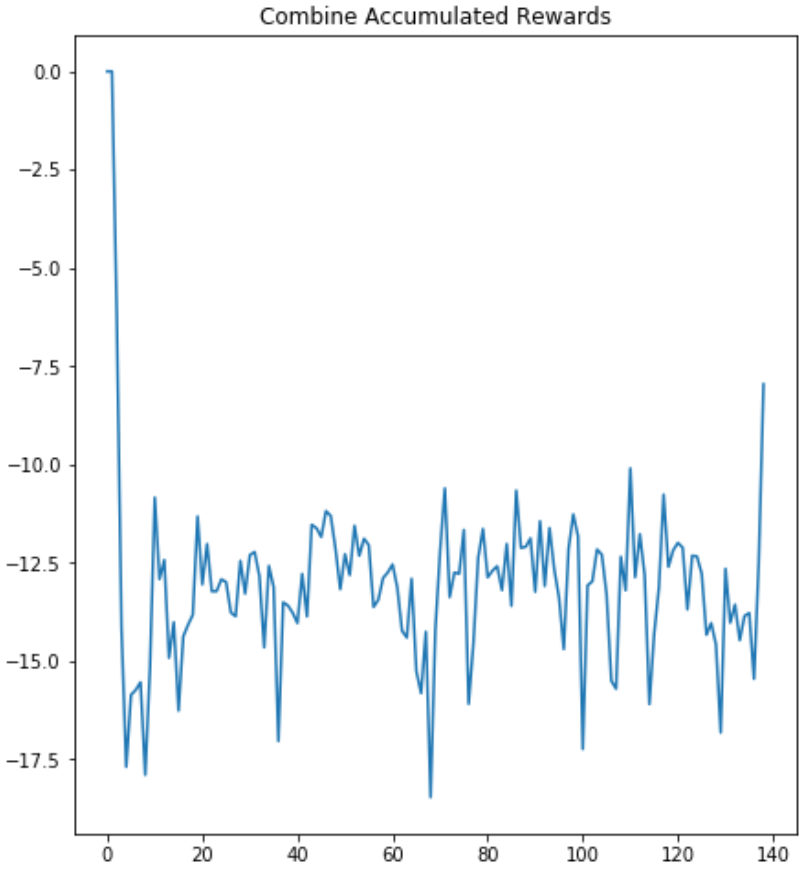
在 Hover 任务中，智能体需要大量的 episode 才能在 185 后开始收敛，Rewards 达到最大，学习及效率不是很理想， 这可能跟折扣因子 gamma 这只为 0.99 有关，考虑后续尝试修改折扣因子来验证这一结果。

2, Landing 任务结果（X 轴 episode， Y 轴 Rewards）：



通过 Landing 的 episode 与 Rewards 数据结果能够看到，当 episode 在 2 后，Rewards 能够比较稳定的收敛，回报达到最高，性能比较稳定。

3, Combine 任务结果（X 轴 episode， Y 轴 Avg_Rewards）：



通过 Combine 的 episode 与 Rewards 数据结果能够看到，当 episode 在 2 后，Rewards 能够稳定在 14 左右，回报达到最高，性能比较稳定，我们注意到 Rewards 在 14 左右徘徊，很大因素可能跟 智能体在 Hover 任务阶段时收敛较慢有关。

5.2 对项目的思考

本项目最困难的部分是 运行 ROS 及完成 Hover 任务。对任务先单个训练智能体，设置合适的参数及奖励函数，选择合适的算法； 由于每个任务是 Reinforcement Learning 要解决的问题：让 agent 学习在一个环境中的如何行为动作(act)， 从而获得最大的奖励值总和(total reward)，这个奖励值一般与 agent 定义的任务目标关联。由于这套飞行任务又是连续性的，需要连续行为的策略学习，因此我们选择使用 DDPG 算法来完成。关于四轴飞行器和智能体的行为，我发现即使智能体在起飞，悬浮，着陆过程中有摇摆行为，但给到的奖励仍然比较高，也就是说奖励无法很全面的评价智能体在全过程中是否稳定运行。

5.3 需要作出的改进

这个项目中，对于 DDPG 算法在连续性动作如起飞，悬停，着陆中使用及推广是可行的，甚至在四轴飞行器的其他连续动作场景中也可以泛化，比如四轴飞行器的急停，稳定旋转，围绕焦点旋转等等。但有些技术需要配合四轴飞行器使用，比如重力感应，四轴升力，平衡感应及计算，障碍物探测等等相关领域。