王者荣耀作业代码

5.1~5.21

内容

- 训练框架原理
- 状态空间、动作空间、奖励
- 示例网络结构
- 代码结构
- 可修改的内容

训练框架原理

- 基于KaiwuDRL框架
- CPU服务器运行多个actor进程和1个model_pool进程，GPU服务器运行1个learner进程和1个model_pool进程
- model_pool负责管理模型参数以及版本

- 详细的通信架构可以参考手册

训练框架原理

- Actor进程负责反复运行对局采集训练数据
  - 每一局开始前从CPU服务器的model_pool中获取模型参数
  - 一方为最新模型，另一方80%取最新模型，20%随机取历史模型
  - 加载模型后与环境交互进行训练，得到一局的(s,a,r)序列
  - 计算出一局的(s,a,adv)序列，批量发送给GPU服务器的ReplayBuffer
- Learner进程负责用ReplayBuffer中收集到的数据进行训练
  - 网络初始参数以及训练过程中产生的新参数都会放入GPU服务器端的model_pool中，进而下发给CPU服务器端的model_pool

动作空间

- 动作空间是层次化的

- 主动作
- 方向
- 目标

动作空间

- 主动作：12个

Button	None	No action	1
	None	No action	1
	Move	Move hero	1
	Normal Attack	Release normal attack	1
	Skill 1	Release 1st skill	1
	Skill 2	Release 2nd skill	1
	Skill 3	Release 3rd skill	1
	Heal Skill	Release heal skill	1
	Chosen Skill	Release the chosen skill	1
	Recall	Start channeling and return to the home fountain after a few seconds if not interrupted	1
	Skill 4	Release 4th skill (Only valid for certain heroes)	1
	Equipment Skill	Release skill provided by certain equipment	1

动作空间

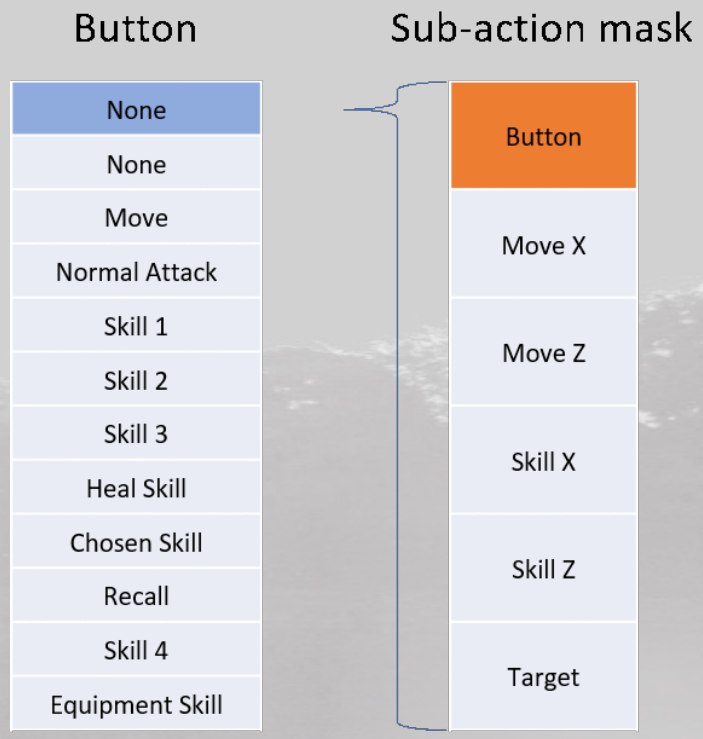
- 移动方向：16+16
- 技能方向：16+16
- 目标：8

Move	Move X	Move direction along X-axis	16
	Move Z	Move direction along Z-axis	16
Skill	Skill X	Skill direction along X-axis	16
	Skill Z	Skill direction along Z-axis	16
Target	None	Empty target	1
	Self	Self player	1
	Enemy	Enemy player	1
	Soldier	4 Nearest soldiers	4
	Tower	Nearest tower	1

- 总动作空间：12+16+16+16+16+8=84

动作空间

- 次动作有效性依赖于主动作



动作空间掩码

- 怎么表达可行动作？
- 12+16+16+16+16+8*12=172

Button	None
	None
	Move
	Normal Attack
	Skill 1
	Skill 2
	Skill 3
	Heal Skill
	Chosen Skill
	Recall
	Skill 4
	Equipment Skill
Move	Move X
	Move Z
Skill	Skill X
	Skill Z
Target	None
	Enemy
	Self
	Soldier
	Tower

Button

None
None
Move
Normal Attack
Skill 1
Skill 2
Skill 3
Heal Skill
Chosen Skill
Recall
Skill 4
Equipment Skill

状态空间

- 环境返回的state_dict包含以下字段
  - observation：809=725+84
  - legal_action：172=12+16+16+16+16+8*12
  - reward：每一项reward字段

状态空间

- observation：809=725+84
- 84是动作空间的大小，表示粗糙的动作掩码（不考虑主动作和目标进行组合的合法性）
- 725=102+133+102+133+14+18*4+18*4+18*2+18*2+25

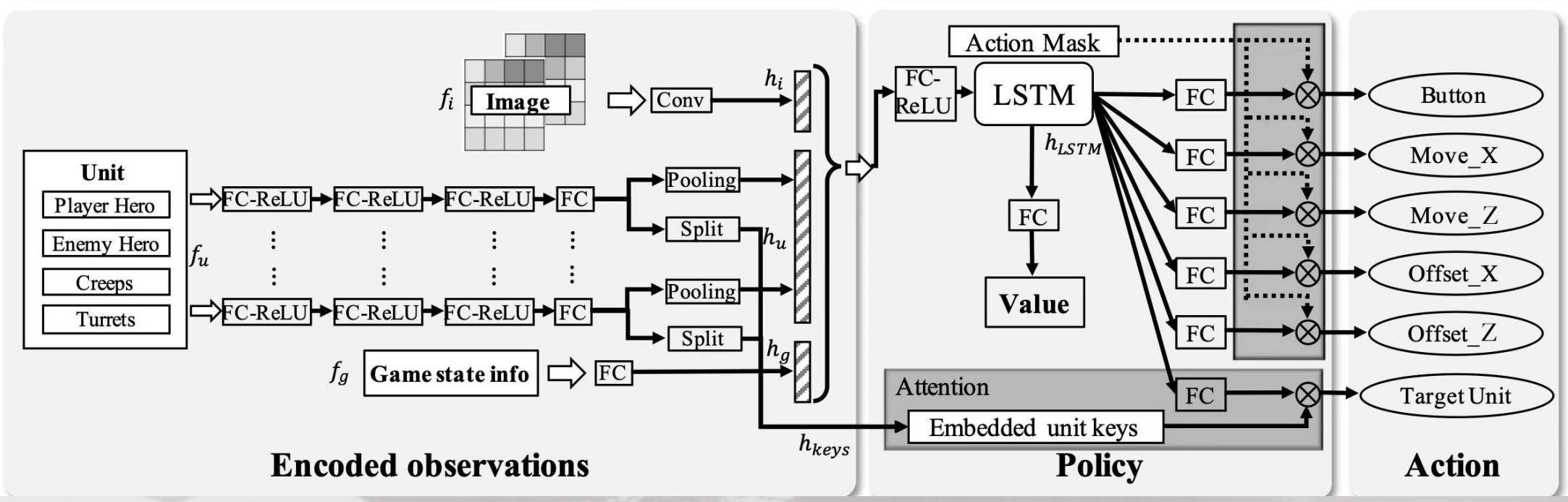
特征区间名	特征维度	举例
Main_camp_hero_state_common_feature	102	我方鲁班血量，位置
Main_camp_hero_private_feature	133	我方鲁班第几次普攻
Enemy_camp_hero_state_common_feature	102	敌方鲁班血量，位置
Enemy_camp_hero_private_feature	133	敌方鲁班第几次普攻
Public_feature	14	敌方小兵是否在我方塔下
Main_camp_soldier_feature	18x4	我方小兵1的血量，位置
Enemy_camp_soldier_feature	18x4	敌方小兵1的血量，位置
Main_camp_organ_feature	18x2	我方防御塔血量，位置
Enemy_camp_organ_feature	18x2	敌方防御塔血量，位置
Global_feature	25	当前游戏时期（前/中/后）

奖励

- 目前环境将奖励分成几部分，分别乘了预设的权重
- 每一项的计算方法可以参考手册
- 权重修改方法：修改app/sgame_1v1/env/feature_process/config.json

reward	权重	类型	描述
hp_point	5.0	dense	the rate of health point of hero
tower_hp_point	1.0	dense	the rate of health point of tower
money (gold)	0.01	dense	the total gold gained
ep_rate	0.75	dense	the rate of mana point
dead	-1.0	sparse	being killed
kill	-0.1	sparse	killing an enemy hero
exp	0.01	dense	the experience gained
last_hit	0.5	sparse	the last hit for creep

示例网络结构


Unit
Player Hero
Enemy Hero
Creeps
Turrets
f_u
FC-ReLU
FC-ReLU
FC-ReLU
FC
f_i
Image
Conv
h_i
Pooling
Split
h_u
Pooling
Split
h_g
f_g
Game state info
FC
h_{keys}
Encoded observations
Action Mask
FC-ReLU
LSTM
h_{LSTM}
FC
Value
FC
FC
FC
FC
FC
FC
Attention
Embedded unit keys
Policy
Button
Move_X
Move_Z
Offset_X
Offset_Z
Target Unit
Action

示例网络结构

- 725=235+235+14+18*4+18*4+18*2+18*2+25
- 特征的不同部分过不同的网络
- 己方英雄235->FC->512->FC->256->FC->128
- 敌方英雄235->FC->512->FC->256->FC->128（96**+32**）
- 局面属性14->FC->64->FC->32->FC->16
- 己方小兵18->FC->64->FC->64->FC->32，四个小兵共用然后max_pool
- 敌方小兵18->FC->64->FC->64->FC->**32**，四个小兵共用然后max_pool
- 己方防御塔18->FC->64->FC->64->FC->32，两个塔共用然后max_pool
- 敌方防御塔18->FC->64->FC->64->FC->**32**，两个塔共用然后max_pool
- 全局信息25不处理
- 得到的特征做拼接128+128+16+32+32+32+32+25=425

示例网络结构

- 425->FC->512->LSTM->512
- Action heads
  - 512->FC->12/16/16/16/16
  - 512->FC->32->**Embed**->8
- Value head
  - 512->FC->64->FC->1
- 8个目标分别用前面处理得到的8个目标特征各截取32维，用这个embedding来加权而不是另外学习一个32*8的权重矩阵

代码结构

- 代码包的根目录下有两个子目录
  - app/sgame_1v1是主要代码文件
    - actor_learner：定义了Agent的逻辑
    - common：包含actor和learner都会调用的神经网络模型代码，以及模型配置参数
    - env：游戏环境的接口包装，负责和gamecore通信
    - tools：一些工具代码，如启动、停止进程以及查看运行结果的脚本
    - docs：代码说明
  - config是配置文件，包含一些配置参数
- 需要理解一些重要API的接口：参考文档

可修改的内容

- reward设计

`app/sgame_1v1/env/feature_process/config.json`里定义了reward的权重，同学们可以通过修改各个reward的权重去训练出玩法风格截然不同的agent，比如提高击杀奖励，agent会变得更好战，提高金钱奖励agent会提高刷钱效率。同学们可以通过消融实验的方法，去测试每个reward对agent训练的影响。但是不同reward之间不是相互独立的，同时修改多个reward的效果不能简单的用单个reward的实验结果进行简单的线性叠加。如何能达到一个相对平衡的点并取得游戏的胜利，是同学们可以思考的方向。

可修改的内容

- 超参数选择

`app/sgame_1v1/common/configs/config.py`和`conf/configue.ini`里包含了很多参数的配置，同学们需要首先学习并了解每个参数的含义以及对训练/推演的影响，判断出哪些是可以去优化的，再利用实验去验证。

超参数的调整一直被认为是深度学习中的"玄学"，同样的算法，同样的参数，针对不同的应用场景都可能会有较大的表现差异。

可修改的内容

- 网络结构设计

`app/sgame_1v1/common/models/model.py`里有对Graph的定义，具体参考`_build_infer_graph`和`_inference`函数，同学们可以在这里进行修改graph的操作；`app/sgame_1v1/actor_learner/game_controller.py`里有对Graph的引用。

- 但样例代码的网络结构已经非常复杂（并且表现相当好）了，推荐先理解已有代码中实现的网络结构，然后可以适当调整网络结构

可修改的内容

- 强化学习算法

`app/sgame_1v1/common/models/model.py`里定义了model，包括loss的计算和inference函数。同学们如果想修改PPO算法，可以从这里入手，但我们只建议进行简单的微调，比如optimizer的调整，或添加小的trick。由于当前框架是针对PPO算法搭建的，所以同学们如果想要尝试PPO以外的算法难度会较大，比如目前不支持sample过程的调整， 涉及到exploration上优化的算法或者value-based的算法都很难实施。