Knights Archers Zombies

Group number:
ST4406-G5

Group member：
ZhangJinwei
Zhu Hengyue
Zhang Ce

1

Related concepts

- Agent: Agent Contains a policy and a set of internal values. It is the object of our learning and can make its own behavior.

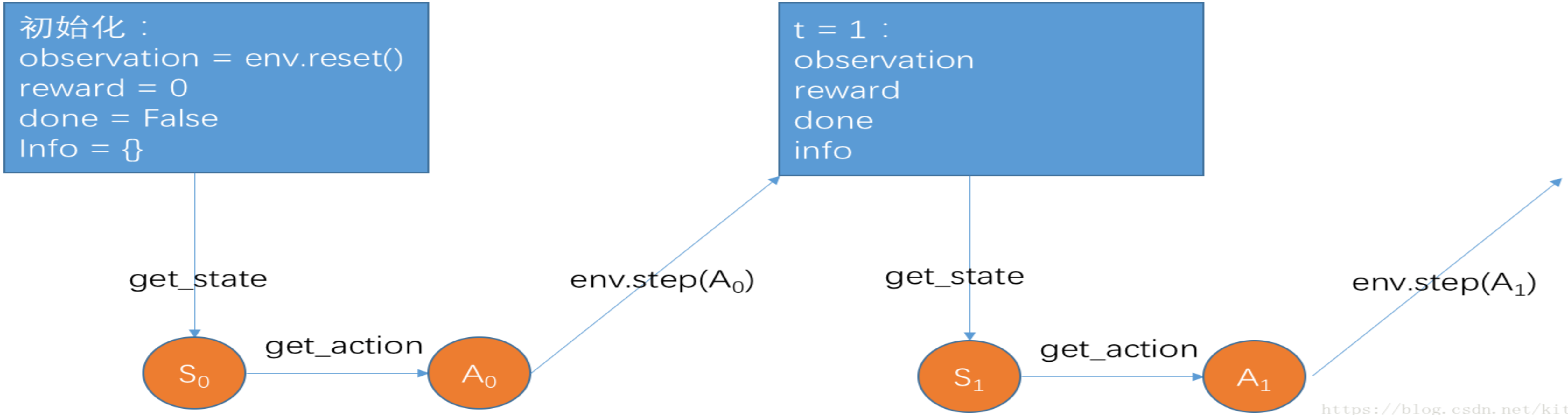
- Environment: Receive action, generate state and reward, including a reward system, which may be random.

- Policy: The most basic function is to receive the Request, and then to provide the corresponding command. The command is the specific processing request.

Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

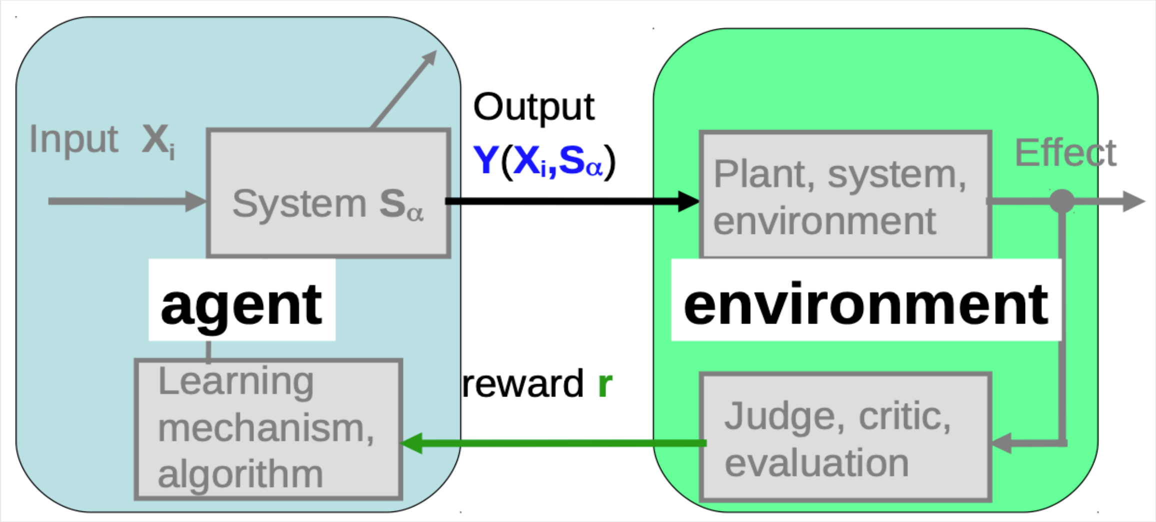
RandomAgent on Pendulum-v0



https://blog.csdn.net/kittyzc

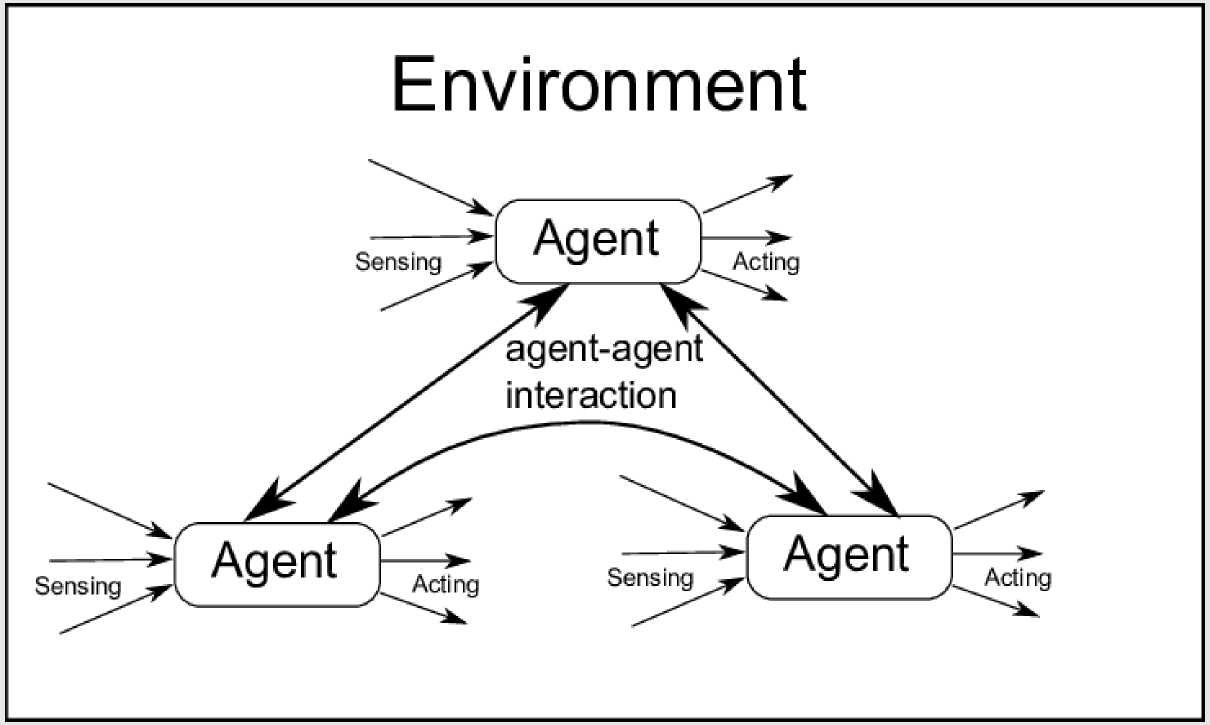
Reinforcement Learning

Reinforcement stems from using machine learning to optimally control an agent in an environment. It works by learning a policy, a function that maps an observation obtained from its environment to an action. Policy functions are typically deep neural networks, which gives rise to the name “deep reinforcement learning.”



Multiagent Reinforcement

In general it’s the same as single agent reinforcement learning, where each agent is trying to learn it’s own policy to optimize its own reward. Using a central policy for all agents is possible, but multiple agents would have to communicate with a central server to compute their actions (which is problematic in most real world scenarios), so in practice decentralized multi-agent reinforcement learning is used.

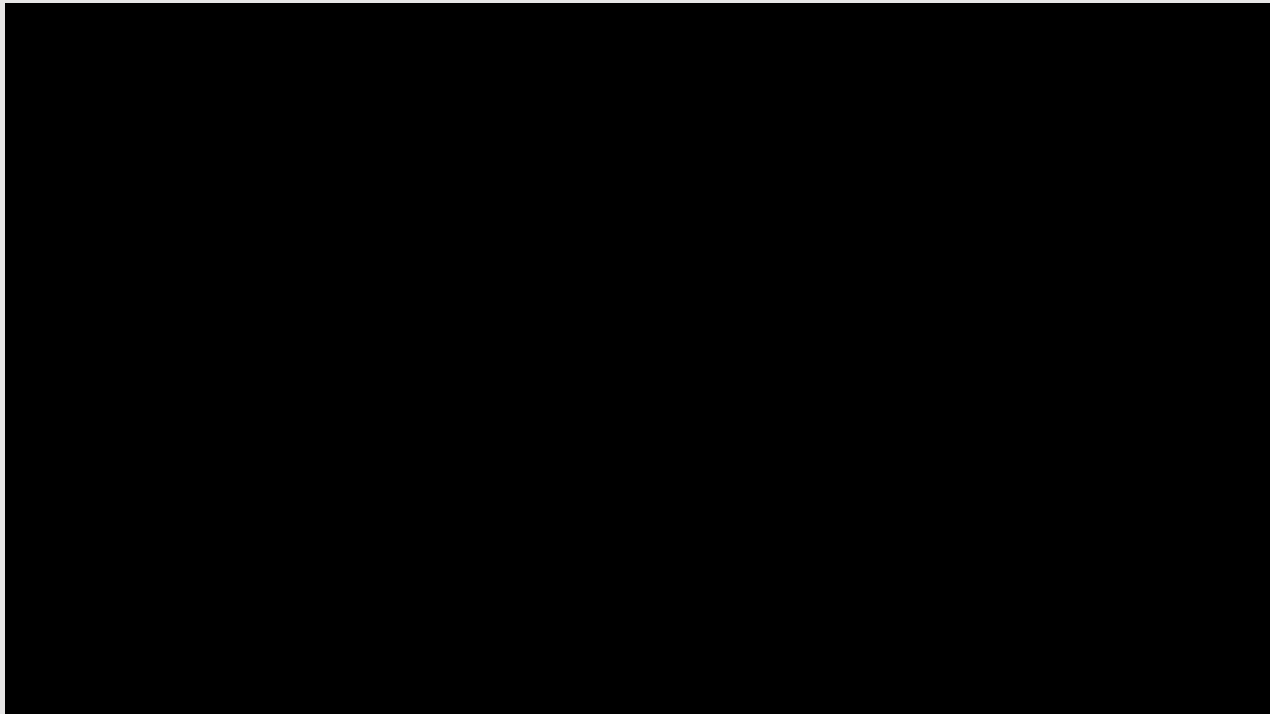


2

Project background

Rules introduction

- Zombies walk from the top border of the screen down to the bottom border in unpredictable paths.
- We control movable knights and archers. A knight is rewarded 1 point when its mace hits and kills a zombie. An archer is rewarded 1 point when one of their arrows hits and kills a zombie.
- The game ends when all agents die or a zombie reaches the bottom screen border.



Our Goal

The goal is for the knights and archers to learn how to kill more zombies before the game is over

3

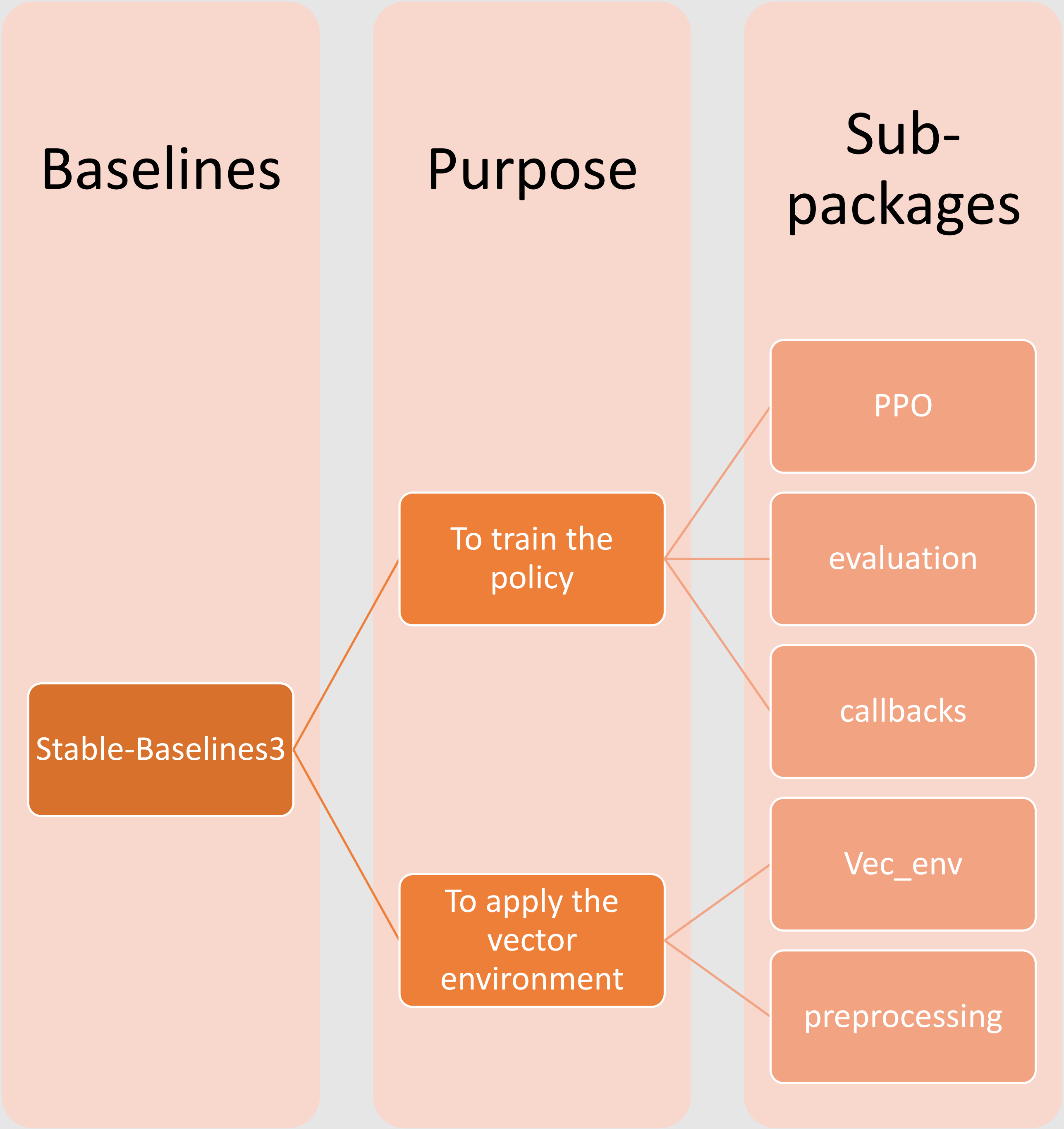
Realization ideas and frame

Realization Ideas

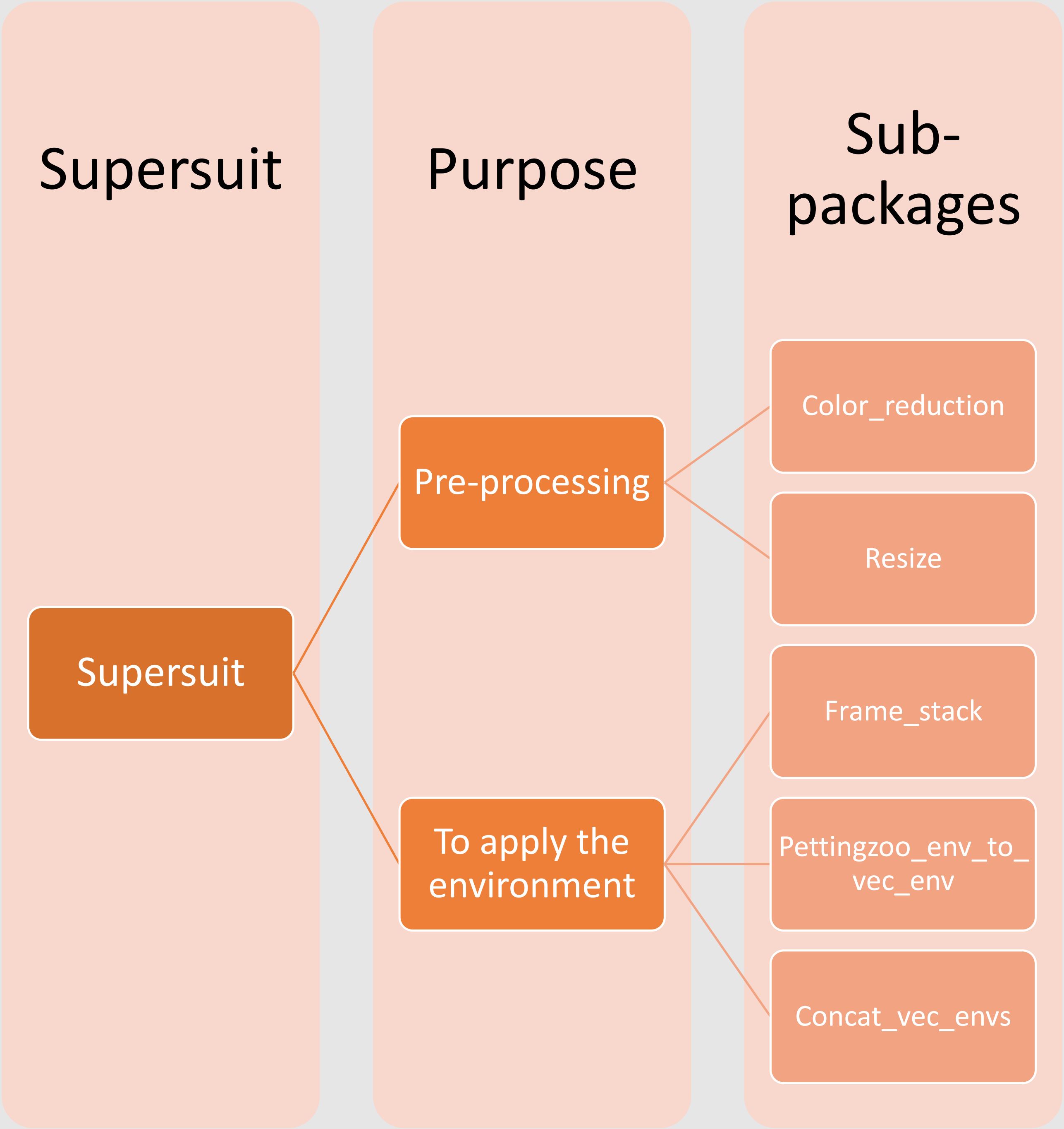
1. Use Stable-Baselines3 for evaluating policy and environment processing

2. Use Supersuit for image simplifying and environment processing

Stable-Baselines3



Supersuit



Supersuit
Purpose
Sub-packages
Supersuit
Pre-processing
To apply the environment
Color_reduction
Resize
Frame_stack
Pettingzoo_env_to_vec_env
Concat_vec_envs

Realization Frame

Environment Setting & Processing

env (parallel)	eval_env(parallel)

Policy Evaluation & Training

PPO	Model.learn

Operating

Set new environment	Classic operation

4

Key algorithm

Proximal Policy Optimization

- Arctor-Critic
- On policy
- Based on TRPO and PG
- Used the probability ratio instead if log probability
- Control the change of policy in each iteration
    - Add an adaptive KL penalty
    - Use CLIP, add an epsilon to control

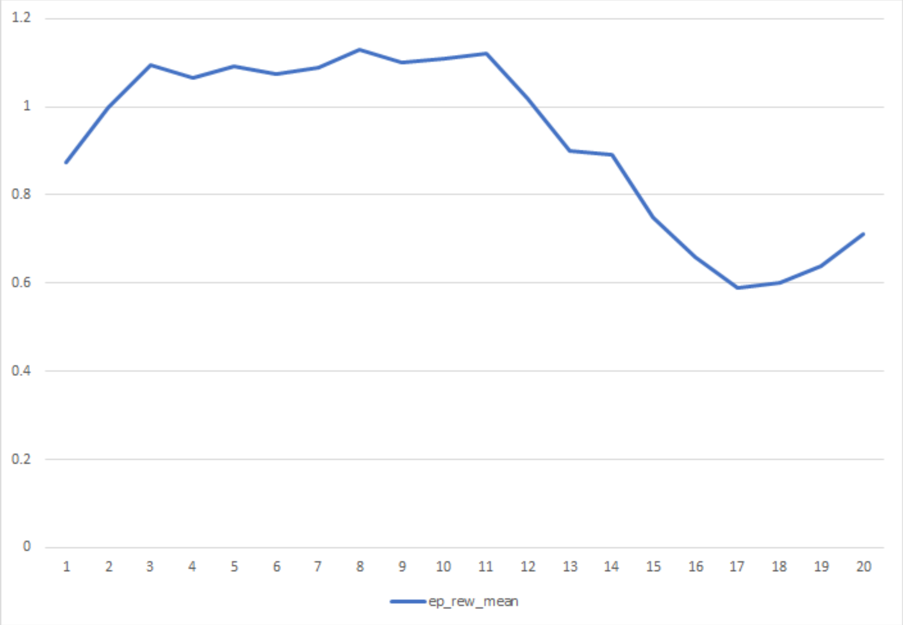
$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\ ]$$

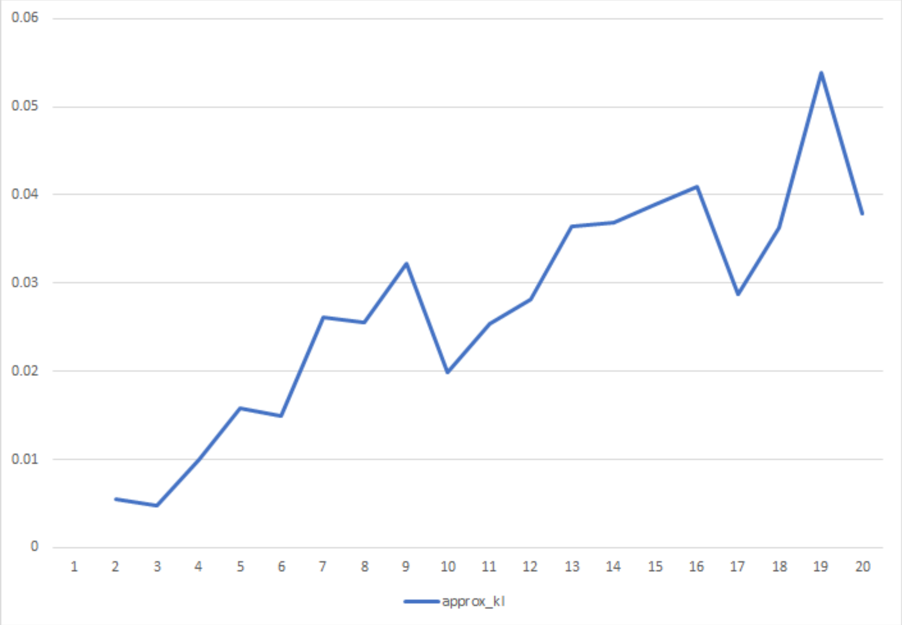
- θ is the policy parameter
- $\hat{E}_t$ denotes the empirical expectation over timesteps
- r_t is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

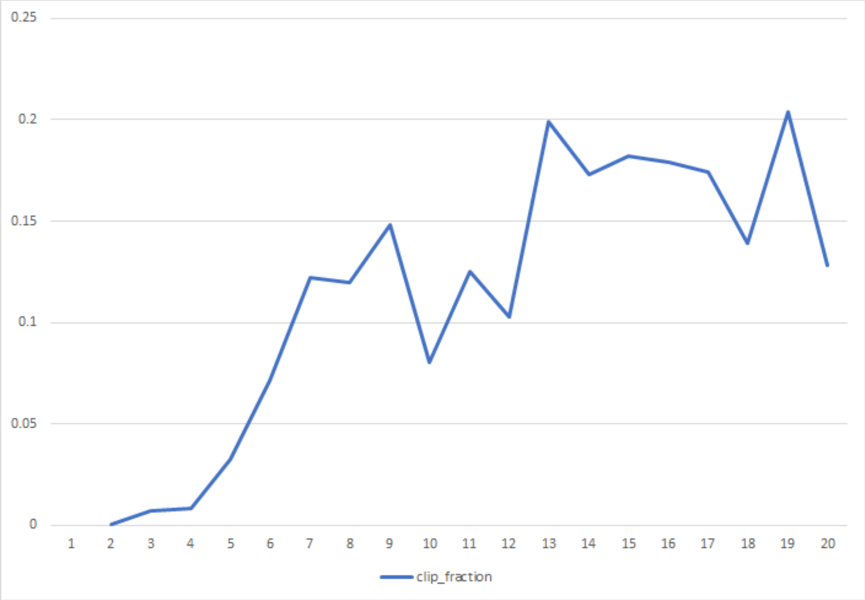
5

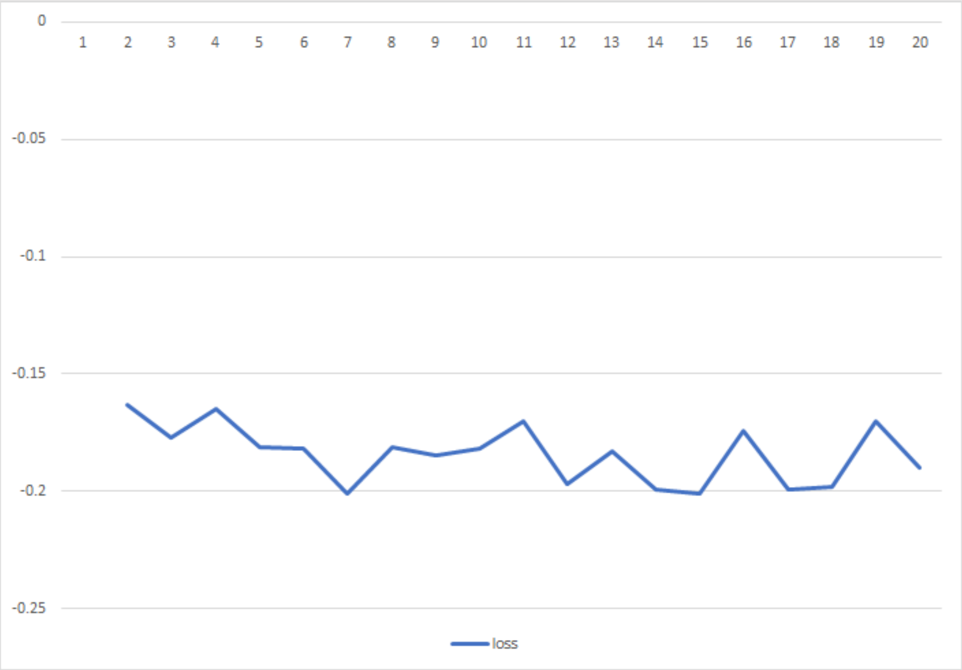
Result display

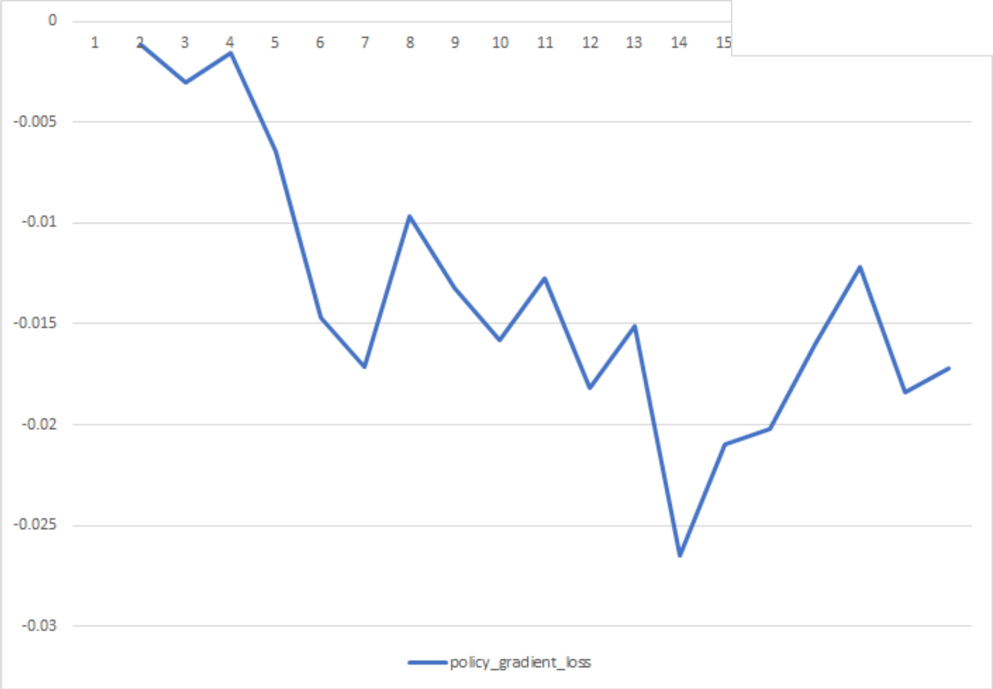
	1	2	3	4	5	6	7	8	9	10
ep_rew_mean	0.875	1	1.09375	1.0652174	1.0925926	1.0735294	1.0897436	1.1304348	1.1	1.11
approx_kl		0.005459972	0.004744264	0.010000747	0.015741933	0.014996057	0.026188482	0.025564905	0.032294594	0.019883174
clip_fraction		0.000488	0.00713	0.00869	0.0324	0.0711	0.122	0.12	0.148	0.0807
entropy_loss		-1.79	-1.79	-1.78	-1.79	-1.77	-1.74	-1.75	-1.76	-1.77
loss		-0.163	-0.177	-0.165	-0.181	-0.182	-0.201	-0.181	-0.185	-0.182
policy_gradient_loss		-0.00116	-0.00304	-0.00157	-0.00644	-0.0147	-0.0171	-0.00964	-0.0132	-0.0158
value_loss		0.0801	0.0589	0.0609	0.0439	0.0415	0.0447	0.0684	0.0555	0.0471
	11	12	13	14	15	16	17	18	19	20
ep_rew_mean	1.12	1.02	0.9	0.89	0.75	0.66	0.59	0.6	0.64	0.71
approx_kl	0.025	0.028212432	0.03647689	0.03683254	0.038915418	0.040916942	0.028782278	0.036273815	0.053855795	0.037858583
clip_fraction	0.125	0.103	0.199	0.173	0.182	0.179	0.174	0.139	0.204	0.128
entropy_loss	-1.77	-1.76	-1.77	-1.75	-1.74	-1.74	-1.75	-1.74	-1.75	-1.75
loss	-0.17	-0.197	-0.183	-0.199	-0.201	-0.174	-0.199	-0.198	-0.17	-0.19
policy_gradient_loss	-0.01	-0.0182	-0.0151	-0.0265	-0.021	-0.0202	-0.016	-0.0122	-0.0184	-0.0172
value_loss	0.048	0.0447	0.0278	0.0128	0.0323	0.0221	0.019	0.0365	0.0369	0.0695

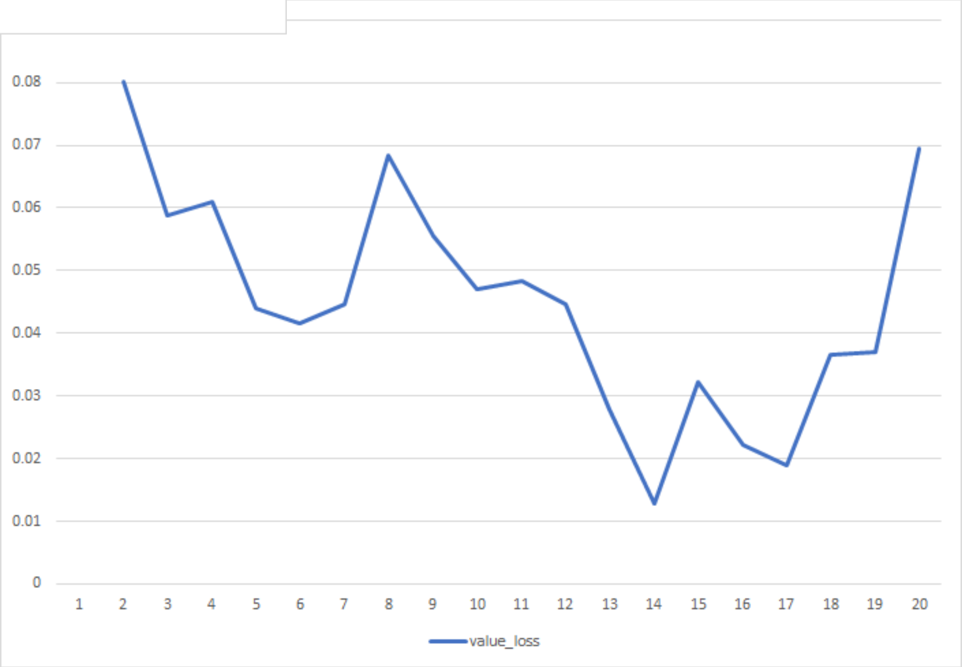

1.2
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
ep_rew_mean


0.06
0.05
0.04
0.03
0.02
0.01
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
approx_kl


0.25
0.2
0.15
0.1
0.05
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
clip_fraction


0
-0.05
-0.1
-0.15
-0.2
-0.25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
loss


0
-0.005
-0.01
-0.015
-0.02
-0.025
-0.03
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
policy_gradient_loss


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
value_loss

6

Possible Improvement

Possible Improvement

Maybe 2 systems with different policy

Two System

System 1:
Use the simple policy, take action by observetion

System 2:
Use PPO to train the model and take the optimized policy

Reference

https://towardsdatascience.com/multi-agent-deep-reinforcement-learning-in-15-lines-of-code-using-pettingzoo-e0b963c0820b

https://github.com/jkterry1/rl_scratch/blob/a5476ce2332e243dafd5bd804c3bac5e7ae176f2/test_evaluation.py

https://github.com/gml16/yare-rl/blob/6d24a596eb870f54d13898dc76c5da2489135190/yare-rl/train.py

THANK YOU