

RLChina第四次习题课参考教程

题目要求:

动手做

1 训练

2 提交

关于环境和代码逻辑

1 环境

2 代码逻辑

正餐: submission.py函数结构讲解

附录: submission代码

写在最后:

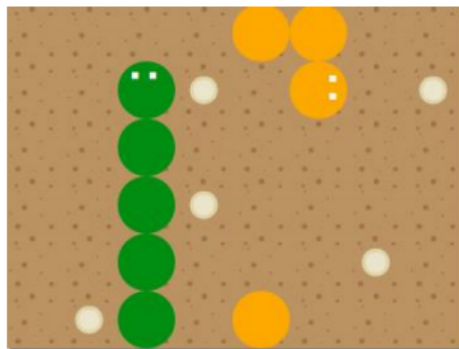
相关文章及资源:

RLChina第四次习题课参考教程

嘻嘻~~ 我又来了~

这次的作业 递进到了单方多智能体决策的环境中

环境场景: <http://www.jidi.ai.cn/snake2p>



- 控制两条蛇，在规定步数（30）内通过吃豆子增加长度
- 若一条蛇头撞上自己或另一条蛇的蛇身会死亡，并随机以长度3重生
- 最终在第30步时，积分=(蛇A长度-3)+(蛇B长度-3)

题目要求:

作业的目的:

理解多智能体算法并且动手训练提交

作业要求

- 训练贪吃蛇(2P)游戏的多智能体合作算法
- 将homework里的 submission.py 填写完整
- 将 submission.py, critic.py, critic_*.pth 提交到及第平台

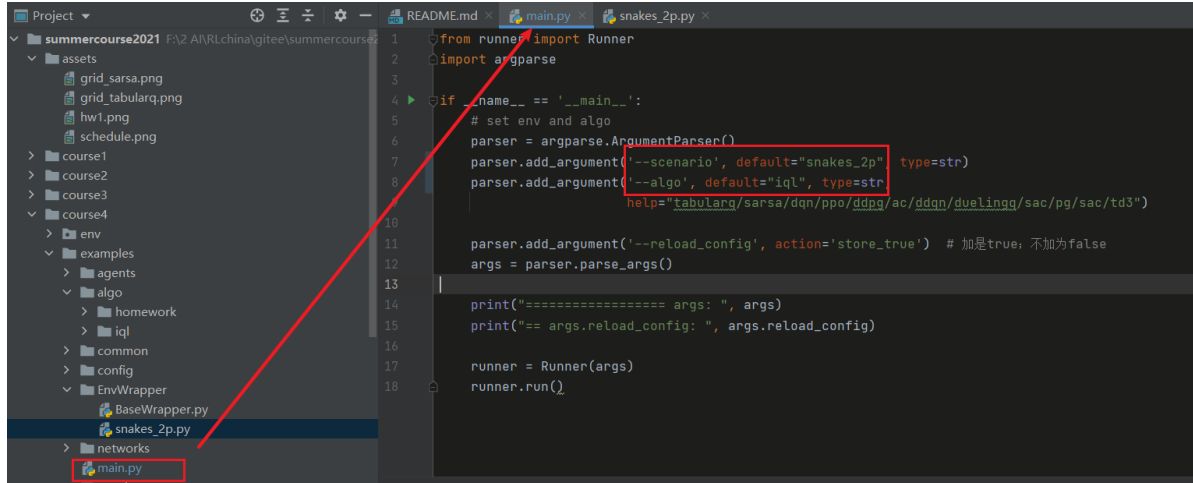
动手做

首先记得更新一下代码到最新版本:

SummerCourse2021: <https://gitee.com/jidiai/summercourse2021>

1 训练

修改main.py函数(也可以用命令行哈)

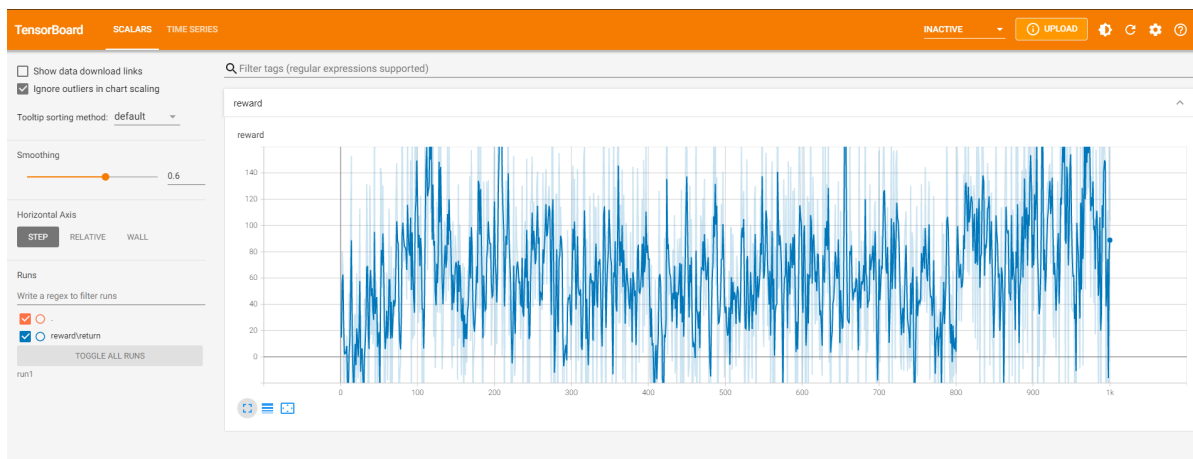


```
1 from runner import Runner
2 import argparse
3
4 if __name__ == '__main__':
5     # set env and algo
6     parser = argparse.ArgumentParser()
7     parser.add_argument('--scenario', default="snakes_2p", type=str)
8     parser.add_argument('--algo', default="iql", type=str)
9     parser.add_argument('--help', help="tabularq/sarsa/dqn/ppo/ddpg/ac/ddqn/duelingg/sac/pg/sac/td3")
10
11 parser.add_argument('--reload_config', action='store_true') # 加是true, 不加为false
12 args = parser.parse_args()
13
14 print("===== args: ", args)
15 print("== args.reload_config: ", args.reload_config)
16
17 runner = Runner(args)
18 runner.run()
```

然后运行就开始训练了, 打印部分输出结果:

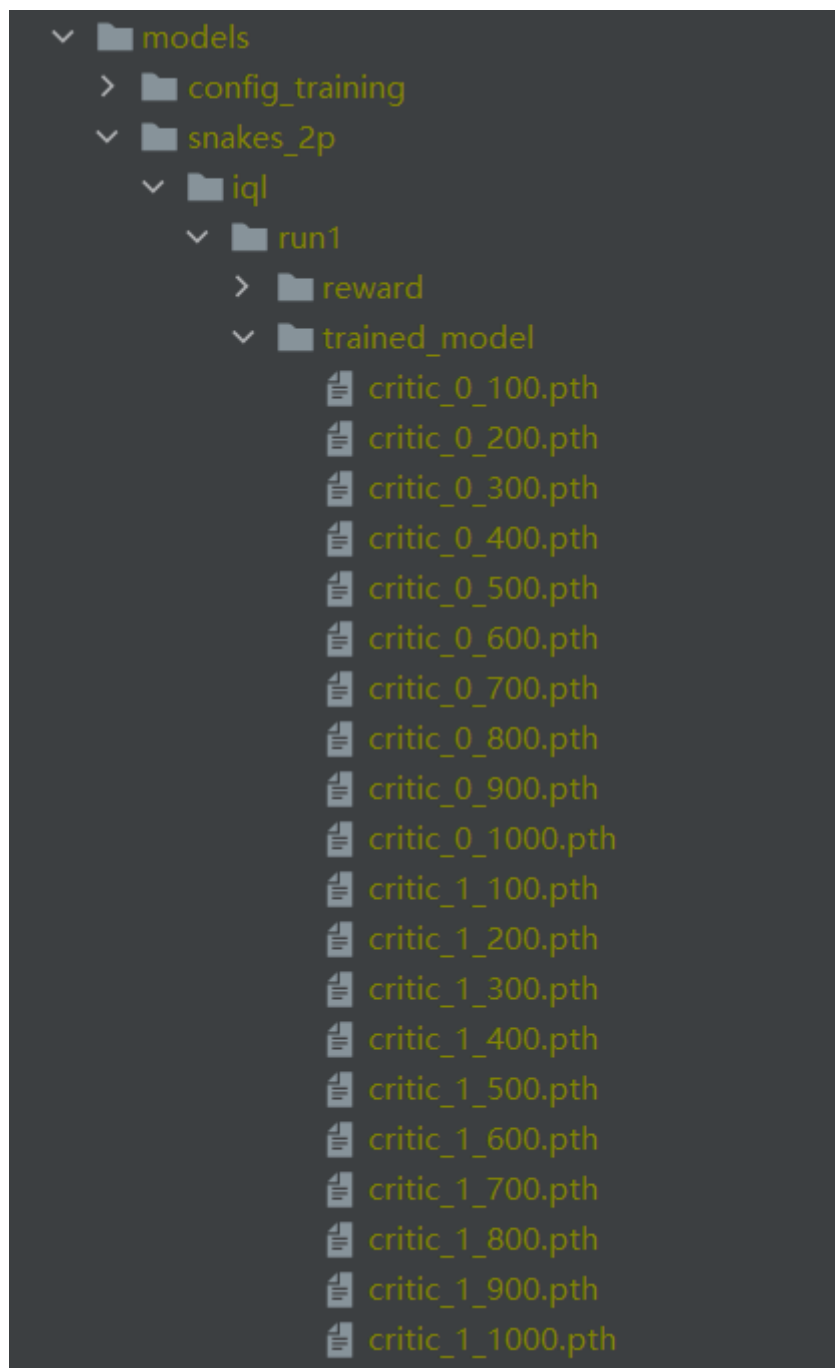
```
1 i_epoch: 994 Gt: 147.80 epsilon: 0.20
2 i_epoch: 995 Gt: -127.00 epsilon: 0.20
3 i_epoch: 996 Gt: 66.97 epsilon: 0.20
4 i_epoch: 997 Gt: 111.52 epsilon: 0.20
5 i_epoch: 998 Gt: -152.23 epsilon: 0.20
6 i_epoch: 999 Gt: 221.84 epsilon: 0.20
7 i_epoch: 1000 Gt: 103.63 epsilon: 0.20
```

tensorboard可视化一下结果(不清楚的同学看习题三的答案哈)



好家伙, 训练这个鬼样子。先不优化, 走一遍作业的流程先

保存的模型长这个样子



这是因为代码逻辑是两条蛇的模型分开保存，这样在读模型的时候会 and 之前有些不同

```
97 done = torch.tensor(transitions["d_0"], dtype=torch.float).squeeze()
98
99 q_eval = self.critic_eval(obs).gather(1, action)
100 q_next = self.critic_target(obs_).detach()
101 q_target = (reward + self.gamma * q_next.max(1)[0] * (1 - done)).view(self.batch_size, 1)
102 loss_fn = nn.MSELoss()
103 loss = loss_fn(q_eval, q_target)
104
105 self.optimizer.zero_grad()
106 loss.backward()
107 self.optimizer.step()
108
109 if self.learn_step_counter % self.target_replace_iter == 0:
110     self.critic_target.load_state_dict(self.critic_eval.state_dict())
111     self.learn_step_counter += 1
112
113 return loss
114
115 def save(self, save_path, episode, id):
116     base_path = os.path.join(save_path, 'trained_model')
117     if not os.path.exists(base_path):
118         os.makedirs(base_path)
119
120     model_critic_path = os.path.join(base_path, "critic_" + str(id) + "_" + str(episode) + ".pth")
121     torch.save(self.critic_eval.state_dict(), model_critic_path)
122
123 def load(self, file):
124     self.critic_eval.load_state_dict(torch.load(file))
125
```

2 提交

这里就是一个主要任务，完成submission.py

这里要加载两个智能体的策略，完事可能还要拼接一下，稍稍有点难度

```
# TODO
class IQL:
    def __init__(self):
        pass

# TODO
def action_from_algo_to_env(joint_action):
    pass

# todo
# Once start to train, u can get saved model. Here we just say it is critic.pth.
critic_net = os.path.dirname(os.path.abspath(__file__)) + '/critic.pth'
agent = IQL()
agent.load(critic_net)

# todo
def my_controller(observation, action_space, is_act_continuous=False):
    obs = observation['obs']
    action = agent.choose_action(obs)
    return action_from_algo_to_env(action)
```

这部分是要主要完成的

但是下面这部分应该也要改

U1S1 这次这个提示给的有点省略太多了。。哭辽

写吧写吧，主要参考一下，

- agent/multiagent.py
- algo/iql
- 还有贪吃蛇3V3的代码也需要用来参考一下

当然，关于action如何被调用，还要看看 runner.py 下的 get_joint_action_eval 函数

昨天刚说直播教大家写(搬运), 结果今天自己就翻车了。呜呜

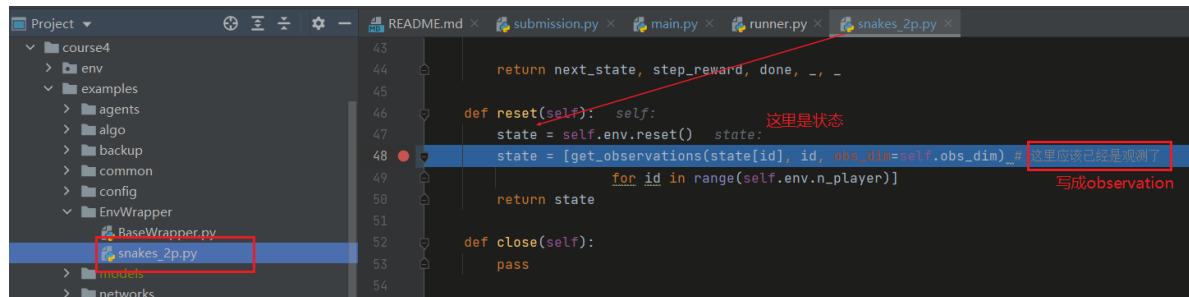
这里理一下代码逻辑, 顺一下

PS: 代码就在文末, 感觉这部分帮助不大就直接划到文末拿代码

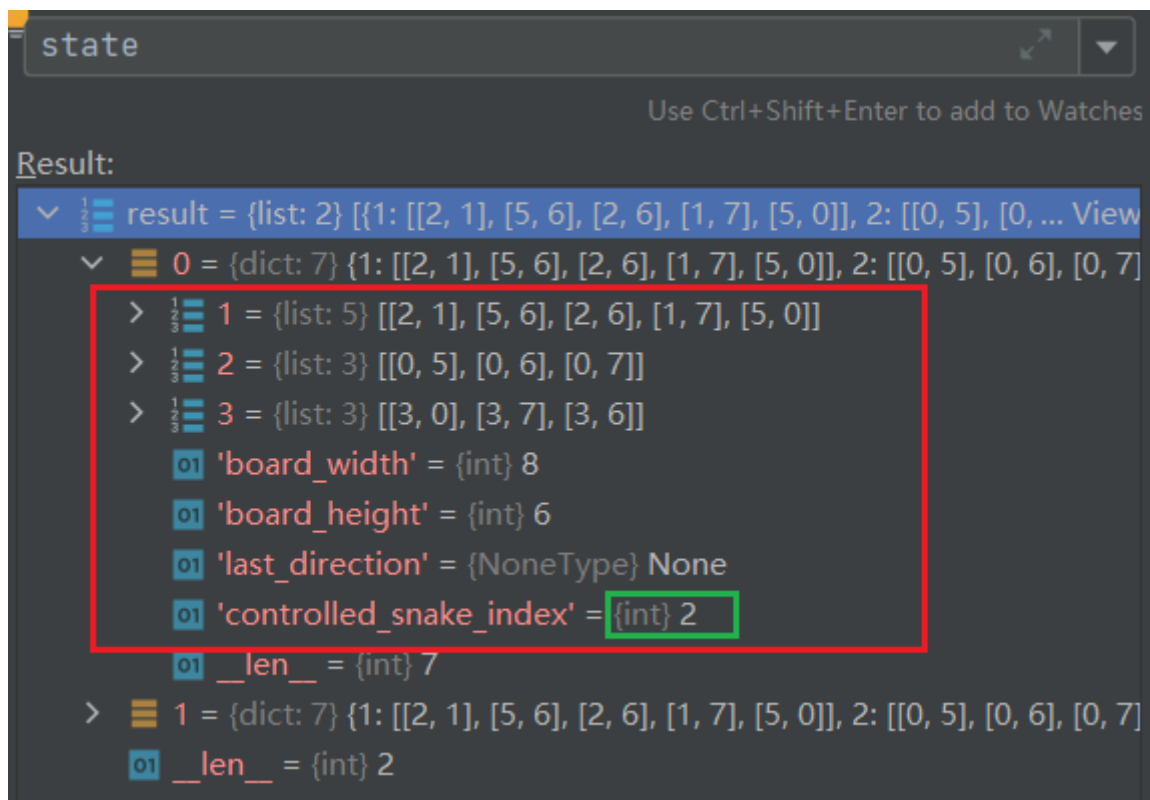
关于环境和代码逻辑

1 环境

状态空间 **state**: 这里源码的命名稍稍有点有歧义



这里打印state看一下, 是两个字典, 每个字典包含如下元素



每个元素的含义正如官网定义:

- **observation**: 一个字典, 字典的键为1至3的正整数和"board_width"、"board_height"、"last_direction"、"controlled_snake_index"。其中1表示豆子, 2至3表示蛇; 字典中的值为一个以[h,w]坐标为元素的列表, 表示豆子的位置或蛇身的位置, 其中h表示距左上角原点的垂直距离, w表示距原点的水平宽度; 在2至3对应的值当中, 列表元素从左至右表示对应蛇的蛇头至蛇尾的位置; "board_width"的值为地图的宽, "board_height"的值为地图的高, "last_direction"的值为上一步各个蛇的方向, "controlled_snake_index"的值为控制蛇的序号;

观测空间 **observation**:

```
43
44     return next_state, step_reward, done, _, _
45
46     def reset(self):
47         self.state = self.env.reset()
48         state = self.get_observations(state[id], id, obs_dim=self.obs_dim)
49         for id in range(self.env.n_player):
50             state[id] = self.get_observations(state[id], id, obs_dim=self.obs_dim)
51         return state
```

打印观测从这里来，就是两个长度为18的列表【所以提交submission里需要有get_observations的代码】

```
> state = {list: 2} [[0.0, 5.0, 0.0, 0.0, 0.0, 2.0, 2.0, 1.0, 5.0, 6.0, 2.0, 6.0, 1.0, 7.0, 5.0, 0.0, 3.0, 0.0], [
> 0 = {list: 18} [0.0, 5.0, 0.0, 0.0, 0.0, 2.0, 2.0, 1.0, 5.0, 6.0, 2.0, 6.0, 1.0, 7.0, 5.0, 0.0, 3.0, 0.0]
> 1 = {list: 18} [3.0, 0.0, 0.0, 0.0, 3.0, 0.0, 2.0, 1.0, 5.0, 6.0, 2.0, 6.0, 1.0, 7.0, 5.0, 0.0, 0.0, 5.0]
```

动作空间 action:就是上下左右四个

但是最终要给到环境的要是one-hot的动作，比如下方的动作 3和0，应该是如下的形式

```
while not self.g_core.is_terminal():
    step += 1
    joint_act = self.get_joint_action_eval(self.env, multi_part_agent_ids, self.policy)
    next_state, reward, done, info_before, info_after = self.env.step(joint_act)

    Evaluate
    Expression:
    joint_act
    Result:
    result = {list: 2} [[[0, 0, 0, 1]], [[1, 0, 0, 0]]]
    0 = {list: 1} [[0, 0, 0, 1]]
    1 = {list: 1} [[1, 0, 0, 0]]
    len = (int) 2
```

2 代码逻辑

这里有同学问到，IQL的网络部分为什么和DQN基本一致。这里简单说明一下。

从逻辑上思考，mutiagent.py 这个文件，本质上就是对IQL(其实我觉得这里用DQN更合适)这个调用了两次

```
14 # 类内agent
15 agent = getattr(agent_file_import, agent_class_name)(args, )
16 return agent
17
18
19 class MultiRLAgents(Baseagent):
20     def __init__(self, args):
21         super(MultiRLAgents, self).__init__(args)
22         self.args = args
23         self.agents = list()
24         self.given_net = Critic(self.args.obs_space, self.args.action_space, self.args.hidden_size)
25
26         for i in range(self.args.n_player):
27             agent_file_name = str("algo." + str(self.args.algo) + "." + str(self.args.algo))
28             agent_file_import = importlib.import_module(agent_file_name)
29             agent_class_name = self.args.algo.upper()
30             if self.args.share_net:
31                 given_net = self.given_net
32             else:
33                 given_net = Critic(self.args.obs_space, self.args.action_space, self.args.hidden_size)
34             # 这个相当于 IQL (args, given_net) (查一下getattr的用法就会清晰了)
35             agent = getattr(agent_file_import, agent_class_name)(args, given_net) # 调用IQL
36             self.agents.append(agent)
37
```

大家可以看到IQL网络这里的一个判断函数

```
20
21 class IQL(object):
22     def __init__(self, args, network):
23
24         self.state_dim = args.obs_space
25         self.action_dim = args.action_space
26
27         self.hidden_size = args.hidden_size
28         self.lr = args.c_lr
29         self.buffer_size = args.buffer_capacity
30         self.batch_size = args.batch_size
31         self.gamma = args.gamma
32         # 对于 实时训练 的网络
33
34         # given_net: true
35         if args.given_net:
36             self.critic_eval = network # 方案1: 参数共享
37         else:
38             self.critic_eval = Critic(self.state_dim, self.action_dim, self.hidden_size) # 方案2: 独立训练
39
40         self.critic_target = Critic(self.state_dim, self.action_dim, self.hidden_size)
41         self.optimizer = optimizer.Adam(self.critic_eval.parameters(), lr=self.lr)
```

所以当这一步判断完之后，这时候后面的代码几乎和DQN完全一致，没有任何区别，这就是大家会觉得混淆的原因

其实个人感觉要是

- class MultiRLAgents 写成 class IQL
- class IQL 写成 class DQN_Share (带选择是否共享网络的DQN)

会直观一些。。可能看法不成熟，但是大家能够理解就好，如果有理解不对的地方，还请大家指出

写到这里，我思路也比较清晰了，感觉可以写submission.py了

但是为了和工程保持一致，别混淆大家，我还是用原来的命名方式哈~

正餐：submission.py函数结构讲解

总共我在代码中分为了几个块：

- IQL算法模块
- 状态转为观测模块
- 加载模型模块
- 以及最后的my_controller

第一块可以直接从 `agent/multiagent.py`、`algo/iql.py` 中把下面三个功能抄出来

- 能初始化一个指定大小的critic网络
- 能加载训练过的网络
- 能用这个网络选择动作

```
15     ''' Block1: IQL Build'''
16     class MultiRLAgents:
17         def __init__(self):...
24
25         def choose_action_to_env(self, observation, id):...
31
32         def action_from_algo_to_env(self, joint_action):...
41
42         def load(self, file_list):
43             for index, agent in enumerate(self.agents):
44                 agent.load(file_list[index])
45
46         # TODO
47         class IQL:
48             def __init__(self):...
57
58             def choose_action(self, observation):...
63
64             def load(self, file):
65                 # pass
66                 self.critic_eval.load_state_dict(torch.load(file))
67         #TODO
```

第二块：直接从 `Envwrapper/snakes_2p.py` 下把这三个函数复制出来

```
67     #TODO
68
69
70     ''' Block2: State to Observations '''
71     def get_surrounding(state, width, height, x, y):...
78
79     def make_grid_map(board_width, board_height, beans_positions:list, snakes_positions:dict):...
89
90     def get_observations(state, id, obs_dim):...
127
```

第三块：加载模型

这里稍稍和第一块的加载方式对应。

其实本质上，只要前面随便初始化好一个对应大小的神经网络，这个参数就可以加载到这个模型中


```

129 # todo
130 ''' Block3: Load your modle '''
131 # Once start to train, u can get saved model. Here we just say it is critic.pth.
132 critic_net_0 = os.path.dirname(os.path.abspath(__file__)) + '/critic_0_1000.pth'
133 critic_net_1 = os.path.dirname(os.path.abspath(__file__)) + '/critic_1_1000.pth'
134 # 不共享网络就加载两套不一样的 共享就加载两套一样的
135
136 critic_list = [critic_net_0, critic_net_1]
137
138 agent = MultiRLAgents()
139 agent.load(critic_list)

```

训练不共享参数可以这么加载

共享参数的话 加载同一套网络就行

第四块：稍微和环境相关，就是大家要知道输入my_controller函数的参数是啥(这里和3V3很像)

- 输入是：observation 一个智能体的输入状态(state)
- 输出是：my_controller函数输入状态对应智能体的 one-hot 动作

具体信息如下

```

143 # todo
144 ''' observation 是一个agent的state 状态信息 包含以下内容 --
145 observation =
146 {1: [[2, 1], [5, 6], [2, 6], [1, 7], [5, 0]],
147  2: [[0, 5], [0, 6], [0, 7]],
148  3: [[3, 0], [3, 7], [3, 6]],
149   'board_width': 8,
150   'board_height': 6,
151   'last_direction': None,
152   'controlled_snake_index': 2
153 }
154 '''
155 # 对一条蛇的观测 输出这条蛇的动作

```

如果明确了这一点，我想后面就比较好组织了

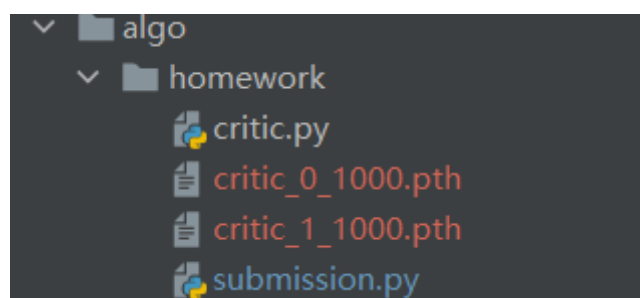
```

155 # 对一条蛇的观测 输出这条蛇的动作
156 def my_controller(observation, action_space, is_act_continuous=False):
157     # obs = observation['obs']
158     obs = observation
159     o_index = obs['controlled_snake_index']
160     o_index -= 2 # 看看observation里面的'controlled_snake_index'和我们蛇的索引正好差2
161
162     obs = get_observations(observation, o_index, 18)
163
164     action_ = agent.choose_action_to_env(obs, o_index)
165
166     return action_

```

感觉大家对这里可能比较困惑，所以多说了几句

整体文件结构：



附录：submission代码

可能有点多，我把这次作业的代码和4次作业的解答一并放到这个网盘中，供大家下载使用

链接:https://pan.baidu.com/s/1xIBGPtHPtC6wbmqVt_jrbw

提取码:4mwa

```
1  ## This is homework.
2  ## Load your model and submit this to Jidi
3
4  import torch
5  import os
6
7  # load critic
8  from pathlib import Path
9  import sys
10 base_dir = Path(__file__).resolve().parent
11 sys.path.append(str(base_dir))
12 from critic import Critic
13 import numpy as np
14
15 ''' Block1: IQL Build'''
16 class MultiRLAgents:
17     def __init__(self):
18         self.agents = list()
19         self.n_player = 2
20
21         for i in range(self.n_player):
22             agent = IQL() # TODO:
23             self.agents.append(agent) # 用不同网络怎么办 -- 还是要拆一下
24
25     def choose_action_to_env(self, observation, id):
26         obs_copy = observation.copy()
27         action_from_algo = self.agents[id].choose_action(obs_copy)
28         action_to_env = self.action_from_algo_to_env(action_from_algo)
29
30         return action_to_env
31
32     def action_from_algo_to_env(self, joint_action):
33         joint_action_ = []
34         for a in range(1):
35             action_a = joint_action
36             each = [0] * 4
37             each[action_a] = 1
38             joint_action_.append(each)
39
40         return joint_action_
41
42     def load(self, file_list):
43         for index, agent in enumerate(self.agents):
44             agent.load(file_list[index])
45
46 # TODO
47 class IQL:
48     def __init__(self):
49         # pass
50         self.state_dim = 18
```

```

51         self.action_dim = 4
52         self.hidden_size = 64
53         # self.given_net_flag = given_net_flag
54         # self.given_net = Critic(self.state_dim, self.action_dim,
self.hidden_size)
55
56         self.critic_eval = Critic(self.state_dim, self.action_dim,
self.hidden_size)
57
58         def choose_action(self, observation):
59             observation = torch.tensor(observation, dtype=torch.float).view(1,
-1)
60             action = torch.argmax(self.critic_eval(observation)).item()
61
62             return action
63
64         def load(self, file):
65             # pass
66             self.critic_eval.load_state_dict(torch.load(file))
67 #TODO
68
69
70 ''' Block2: State to Observations '''
71 def get_surrounding(state, width, height, x, y):
72     surrounding = [state[(y - 1) % height][x], # up
73                   state[(y + 1) % height][x], # down
74                   state[y][(x - 1) % width], # left
75                   state[y][(x + 1) % width]] # right
76
77     return surrounding
78
79 def make_grid_map(board_width, board_height, beans_positions:list,
snakes_positions:dict):
80     snakes_map = [[[0] for _ in range(board_width)] for _ in
range(board_height)]
81     for index, pos in snakes_positions.items():
82         for p in pos:
83             snakes_map[p[0]][p[1]][0] = index
84
85     for bean in beans_positions:
86         snakes_map[bean[0]][bean[1]][0] = 1
87
88     return snakes_map
89
90 def get_observations(state, id, obs_dim):
91     state_copy = state.copy()
92     board_width = state_copy['board_width']
93     board_height = state_copy['board_height']
94     beans_positions = state_copy[1]
95     snakes_positions = {key: state_copy[key] for key in state_copy.keys() &
{2, 3, 4, 5, 6}}
96     snakes_positions_list = []
97     for key, value in snakes_positions.items():
98         snakes_positions_list.append(value)
99     snake_map = make_grid_map(board_width, board_height, beans_positions,
snakes_positions)
100     state = np.array(snake_map)
101     state = np.squeeze(snake_map, axis=2)

```

```

102
103     observations = np.zeros((1, obs_dim)) # todo
104     snakes_position = np.array(snakes_positions_list, dtype=object)
105     beans_position = np.array(beans_positions, dtype=object).flatten()
106     agents_index = [id]
107     for i, element in enumerate(agents_index):
108         # # self head position
109         observations[i][:2] = snakes_positions_list[element][0][:]
110
111         # head surroundings
112         head_x = snakes_positions_list[element][0][1]
113         head_y = snakes_positions_list[element][0][0]
114
115         head_surrounding = get_surrounding(state, board_width,
116 board_height, head_x, head_y)
117         observations[i][2:6] = head_surrounding[:]
118
119         # beans positions
120         observations[i][6:16] = beans_position[:]
121
122         # other snake positions # todo: to check
123         snake_heads = np.array([snake[0] for snake in snakes_position])
124         snake_heads = np.delete(snake_heads, element, 0)
125         observations[i][16:] = snake_heads.flatten()[:]
126
127     return observations.squeeze().tolist()
128
129 # todo
130 ''' Block3: Load your modle '''
131 # Once start to train, u can get saved model. Here we just say it is
132 critic.pth.
133 critic_net_0 = os.path.dirname(os.path.abspath(__file__)) +
134 '/critic_0_1000.pth'
135 critic_net_1 = os.path.dirname(os.path.abspath(__file__)) +
136 '/critic_1_1000.pth'
137 # 不共享网络就加载两套不一样的 共享就加载两套一样的
138 critic_list = [critic_net_0, critic_net_1]
139
140 agent = MultiRLAgents()
141 agent.load(critic_list)
142
143 n_player = 2
144 # todo
145 ''' observation 是一个agent的state 状态信息 包含以下内容 --
146 observation =
147 {1: [[2, 1], [5, 6], [2, 6], [1, 7], [5, 0]],
148 2: [[0, 5], [0, 6], [0, 7]],
149 3: [[3, 0], [3, 7], [3, 6]],
150     'board_width': 8,
151     'board_height': 6,
152     'last_direction': None,
153     'controlled_snake_index': 2
154 }
155 '''
156 # 对一条蛇的观测 输出这条蛇的动作

```

```

156 def my_controller(observation, action_space, is_act_continuous=False):
157     # obs = observation['obs']
158     obs = observation
159     o_index = obs['controlled_snake_index']
160     o_index -= 2 # 看看observation里面的'controlled_snake_index'和我们蛇的索引正
    好差2
161
162     obs = get_observations(observation, o_index, 18)
163
164     action_ = agent.choose_action_to_env(obs, o_index)
165
166     return action_

```

写在最后：

其实我之前写代码的经历也不算多，这几次的作业，虽然能稍稍看懂一些，但是在代码调试、算法优化上很麻爪，可以看到baseline的分数也不是很高。还有3V3比赛，说实话自己训练完baseline连进一步优化的方向都很迷茫

>	贪吃蛇(2P)	iq12	2.20	2021-08-20 19:39:38	通过	 
---	---------	------	------	---------------------	----	---

特别希望分数高的同学能够把自己的优化的方法，思考，甚至是代码，能分享一下。我想这些远比我这写的花拳绣腿有价值。

所以我在论道发起了关于第四次作业优化的问题，期待各位大佬们的回答

及第

金榜

科目

秘籍

擂台

论道

赶考

[置顶] 【习题课】习题课第四天20210819

jidi_admin

2021-08-20 11:56:26

46

1

0

[置顶] ==及第算法库jidiAlgoV1.0==

jidi_admin

2021-08-03 11:49:31

1180

4

0

[置顶] 提交说明 Submit Instructions

jidi_admin

2021-08-03 10:53:55

1566

3

3

RLChina第四次习题课 -- snakes_2p 优化方法

HandsomeWu

2021-08-20 20:41:48

0

0

0

+

新主题

OK ,That's all

特别鸣谢：

小桐姐对问题的耐心解答和指导 -- 及第ID: [Atongmu]

刘子毅同学提供的一版submission.py代码 --- 及第ID:[ZiyiBird]

作者：HandsomeWu(公众号同步)



PS: 天投票的同学还是蛮多的，超过了预期。我后面看情况在B站开10几分钟的小直播，和大家分享下写 submission的思路和一点点pycharm的调试技巧，时间就发在号里叭 不然也联系不到大家 ~

也欢迎关注公众号：**RLCN** 在后台提问：



相关文章及资源:

网盘链接:

链接:https://pan.baidu.com/s/1xIBGPtHPtC6wbmqVt_jrbw

提取码:4mwa

RLChina第一次习题课参考教程

RLChina第二次习题课参考教程

RLChina第三次习题课参考教程

RLChina2021-习题课3 -- 林舒 中科院自动化研究所