

机器人学-强化学习大作业教程

环境安装

1. 首先先确保已经安装第一次大作业所用的环境 `env_dofbot2025`
2. 解压缩作业代码文件 `Grasp_DQN_TODO`
3. pip安装需要的包

代码块

```
1 pip install -r requirements.txt -i  
https://mirrors.cloud.tencent.com/pypi/simple/
```

```
1 gymnasium==0.29.1  
2 tyro  
3 tensorboard  
4
```

4. 尝试import这三个包，如果没有报错，说明已经成功安装。
5. 环境安装完成，可以开始完善代码了！

大作业-代码完善指南

RL仿真环境完善

补全`panda_env.py`中缺失的部分

1. 奖励函数

```
258 # TODO: 完善reward function  
259 def _get_reward(self): 1个用法  
260     obs = self._get_obs_dict()  
261     info = self._get_info()  
262  
263     reward = 0  
264     return reward
```

2. 定义动作空间和观测空间

- a. 环境初始化时定义

```

76         # TODO: observation space
77         # if obs_mode == "state": ## 训练模式下使用
78         #     self.observation_space =
79         # elif obs_mode == "state_dict": ## 字典形式，方便读取数据
80         #     self.observation_space =
81
82         # TODO: action space
83         # self.action_space =

```

b. 从环境中获取observation信息

```

163     def _get_obs_dict(self): 3 用法
164         Observation = self._panda.getObservation()
165
166         # TODO: add suitable observations here
167
168         return Observation

```

3. step函数

```

170         ## discrete action: -dx, dx, -dy, dy, -dz, dz, static
171         def step(self, action): 3 个用法 (3 个动态)
172
173             # TODO: Define suitable realAction here
174             # self.realAction = np.array([dx, dy, dz, 0.04])
175
176
177             if self.terminated:
178                 self.realAction = np.array([0, 0, 0, 0])
179                 self._panda.applyAction(self.realAction)
180                 p.stepSimulation()
181                 if self.render_mode == "human":
182                     time.sleep(self._timeStep)

```

此时，一个完整的gym形式的强化学习仿真环境搭建成功。

可以运行`test_gym.py`，验证仿真环境的设置合理性。

1. 打印observation space和action space

```

print(env.observation_space)
print(env.action_space)

```

2. 输入action，验证动作设计的合理性

3. 执行动作后，打印reward，思考reward设计的合理性

```

observation, reward, terminated, truncated, info = env.step(action)
print("reward: ", reward)

```

训练

运行`dqn_train.py`，进行训练

运行命令示例：`python dqn_train.py --run-name="dqn_demo" --total_timesteps=15_000_000`

- `run-name`：保存的模型文件的文件目录名
- `total_timesteps`：模型训练过程中，与环境交互的总步数。助教设置训练的时候设置了20M，训练了6h左右。实际上13M的时候就收敛了。大家可以根据实际情况自行调整。

ps: `dqn_train.py`中有许多可调参数，本次大作业不要求调整，有兴趣的同学可以自行了解

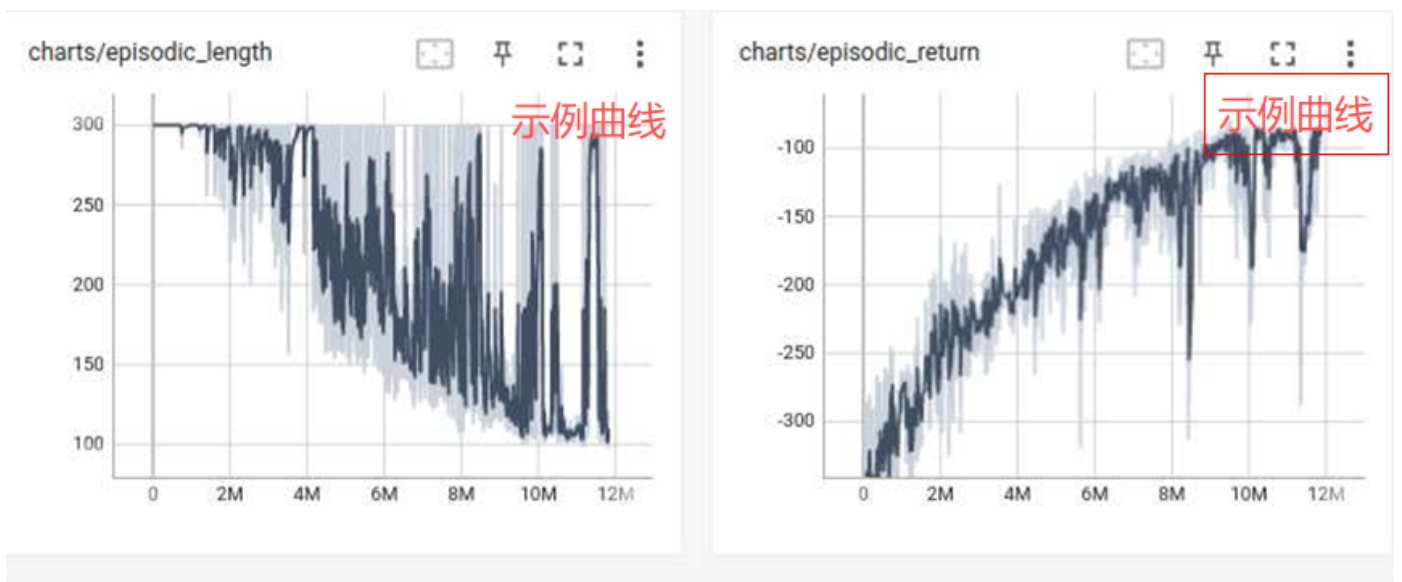
```
# Algorithm specific arguments
env_id: str = "PandaEnv-v1"
    """the id of the environment"""
max_episode_steps: int = 300
total_timesteps: int = 2_000_000
    """total timesteps of the experiments"""
learning_rate: float = 2.5e-4
    """the learning rate of the optimizer"""
num_envs: int = 1
    """the number of parallel game environments"""
buffer_size: int = 10000
    """the replay memory buffer size"""
gamma: float = 0.99
```

评估

1. 通过查看训练曲线评估

由于训练需要的时间较长，在训练过程中可以查看训练曲线，分析训练趋势是否正确。如果感觉到不太对劲可以尽早kill掉，重新调整下仿真环境的设置。（及时止损

- tensorboard查看：`tensorboard --logdir= "runs/<run_name>"`
- 怎么分析训练曲线：一般看看`episodic_return`有没有在增加以及`episodic_length`（是否没等到episode的最大长度，就已经任务成功，结束episode了）



2. 通过加载训练模型，可视化训练结果

- 模型保存形式: `dqn_train_xxx.cleanrl_model`
- 模型加载:

代码块

```
1 model = Model(envs).to(device)
2 model.load_state_dict(torch.load(model_path, map_location=device))
3 model.eval()
```

- evaluate代码: `python dqn_eval.py`
 - 注意要把加载的模型改成你自己训练好的

```
64 if __name__ == "__main__":
65     from dqn_train import QNetwork
66
67     evaluate(
68         model_path: "runs/dqn_demo/dqn_train_final.cleanrl_model",
69         make_env,
70         env_id: "PandaEnv-v1",
71         eval_episodes=1,
72         run_name=f"eval/dqn_demo",
73         Model=QNetwork,
74         device="cpu",
75         capture_video=False,
76     )
```