初识 gym

陶云浩 2017201985

第一次实验为了解 OpenAI Gym 的功能和组成，安装相应 python 库，搭建开发测试环境，并通过运行实例对 Gym 有更进一步的了解。

选用 OpenAI Gym 自带的 examples/agents/random_agent.py 做运行测试。

对 CartPoleEnv 环境源代码及简介进行学习，对其功能有初步了解：该环境模拟在一个可左右移动的小车上放置一根杆子，杆子随时间随机左右摆动。要求通过实时输入 0、1 操纵小车左右移动确保杆子倾斜角度和小车位置维持在一定范围内，坚持时间越久得分越高。相关参数设置如下：

```
Observation:
    Type: Box(4)
    Num     Observation               Min         Max
    0       Cart Position             -4.8          4.8
    1       Cart Velocity             -Inf          Inf
    2       Pole Angle                 -24 deg      24 deg
    3       Pole Velocity At Tip      -Inf          Inf

Actions:
    Type: Discrete(2)
    Num     Action
    0       Push cart to the left
    1       Push cart to the right
```

运行下列测试代码：

Editor – C:\Users\14744\Desktop\人工智能导论\test.py

trans.py | test.py

```python
import argparse
import sys

import gym
from gym import wrappers, logger

class RandomAgent(object):
    def __init__(self, action_space):
        self.action_space = action_space

    def act(self, observation, reward, done):
        return self.action_space.sample()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description=None)
    parser.add_argument('env_id', nargs='?', default='CartPole-v0', help='Select the environment to run')
    args = parser.parse_args()

    # You can set the level to logger.DEBUG or logger.WARN if you
    # want to change the amount of output.
    logger.set_level(logger.INFO)

    env = gym.make(args.env_id)

    # You provide the directory to write to (can be an existing
    # directory, including one with existing data -- all monitor files
    # will be namespaced). You can also dump to a tempdir if you'd
    # like: tempfile.mkdtemp().
    outdir = '/tmp/random-agent-results'
    env = wrappers.Monitor(env, directory=outdir, video_callable=False , force=True)
    env.seed(0)
    agent = RandomAgent(env.action_space)

    episode_count = 100
    reward = 0
    done = False

    for i in range(episode_count):
        ob = env.reset()
        while True:
            action = agent.act(ob, reward, done)
            ob, reward, done, _ = env.step(action)
            if done:
                break
            # Note there's no env.render() here. But the environment still can open window and
            # render if asked by env.monitor: it calls env.render('rgb_array') to record video.
            # Video is not recorded every episode, see capped_cubic_video_schedule for details.

    # Close the env and write monitor result info to disk
    env.close()
```

代码分析，前半段代码规定了输出定位、Gym 环境选择等。在 for 循环中重复了 100 次对设计的算法进行测试。每次 while 循环前将 env 重置，然后重复调用 env.step(action)进行运行测试。

得出结果如下：

```
In [5]: runfile('C:/Users/14744/Desktop/人工智能导论/test.py', wdir='C:/Users/14744/Desktop/人工智能导论')
INFO: Making new env: CartPole-v0
INFO: Clearing 2 monitor files from previous run (because force=True was provided)
INFO: Finished writing results. You can upload them to the scoreboard via gym.upload('C:\\tmp\\random-agent-results')

In [6]:
```

文件中保存的结果如下：

{"initial_reset_timestamp": 1572026639.695147, "timestamps": [1572026639.6961443, 1572026639.6961443, 1572026639.6961443,
1572026639.6971426, 1572026639.6971426, 1572026639.6971426, 1572026639.6981385, 1572026639.6981385, 1572026639.6981385,
1572026639.6981385, 1572026639.6991355, 1572026639.6991355, 1572026639.6991355, 1572026639.6991355, 1572026639.7001328,
1572026639.7001328, 1572026639.7011304, 1572026639.7011304, 1572026639.7021308, 1572026639.7021308, 1572026639.7031536,
1572026639.7031536, 1572026639.7031536, 1572026639.7031536, 1572026639.7031536, 1572026639.7041433, 1572026639.7041433,
1572026639.7041433, 1572026639.7041433, 1572026639.70512, 1572026639.70512, 1572026639.70512, 1572026639.70512, 1572026639.70512,
1572026639.7061179, 1572026639.7061179, 1572026639.7061179, 1572026639.7061179, 1572026639.7071145, 1572026639.7071145,
1572026639.7071145, 1572026639.7071145, 1572026639.708123, 1572026639.708123, 1572026639.7091117, 1572026639.7091117,
1572026639.7091117, 1572026639.7091117, 1572026639.7091117, 1572026639.71013, 1572026639.71013, 1572026639.7111049, 1572026639.7111049,
1572026639.7111049, 1572026639.7111049, 1572026639.7121017, 1572026639.7121017, 1572026639.7121017, 1572026639.7131195,
1572026639.7131195, 1572026639.7131195, 1572026639.7131195, 1572026639.7131195, 1572026639.7141206, 1572026639.7141206,
1572026639.7141206, 1572026639.7141206, 1572026639.7141206, 1572026639.715093, 1572026639.715093, 1572026639.715093, 1572026639.715093,
1572026639.7160907, 1572026639.7160907, 1572026639.7160907, 1572026639.7160907, 1572026639.7171094, 1572026639.7171094,
1572026639.7171094, 1572026639.7171094, 1572026639.7180855, 1572026639.7180855, 1572026639.7180855, 1572026639.7180855,
1572026639.719104, 1572026639.719104, 1572026639.720101, 1572026639.720101, 1572026639.720101, 1572026639.720101, 1572026639.7210789,
1572026639.7210789, 1572026639.7210789, 1572026639.7220762, 1572026639.7220762, 1572026639.7220762, 1572026639.7220762,
1572026639.7230935, 1572026639.7230935, 1572026639.7230935], "episode_lengths": [10, 13, 25, 24, 43, 10, 13, 27, 23, 15, 23, 22, 12,
20, 24, 12, 67, 13, 57, 9, 15, 19, 16, 13, 21, 23, 14, 16, 12, 13, 21, 19, 12, 16, 9, 11, 13, 16, 27, 21, 15, 16, 13, 29, 18, 11, 17,
24, 13, 14, 15, 15, 21, 35, 16, 12, 17, 37, 17, 29, 11, 15, 9, 11, 11, 22, 30, 13, 18, 19, 17, 19, 20, 15, 18, 15, 26, 18, 26, 19, 28,
14, 21, 16, 28, 10, 25, 18, 31, 12, 17, 23, 32, 28, 12, 13, 12, 22, 11, 15], "episode_rewards": [10.0, 13.0, 25.0, 24.0, 43.0, 10.0,
13.0, 27.0, 23.0, 15.0, 23.0, 22.0, 12.0, 20.0, 24.0, 12.0, 67.0, 13.0, 57.0, 9.0, 15.0, 19.0, 16.0, 13.0, 21.0, 23.0, 14.0, 16.0,
12.0, 13.0, 21.0, 19.0, 12.0, 16.0, 9.0, 11.0, 13.0, 16.0, 27.0, 21.0, 15.0, 16.0, 13.0, 29.0, 18.0, 11.0, 17.0, 24.0, 13.0, 14.0,
15.0, 15.0, 21.0, 35.0, 16.0, 12.0, 17.0, 37.0, 17.0, 29.0, 11.0, 15.0, 9.0, 11.0, 11.0, 22.0, 30.0, 13.0, 18.0, 19.0, 17.0, 19.0,
20.0, 15.0, 18.0, 15.0, 26.0, 18.0, 26.0, 19.0, 28.0, 14.0, 21.0, 16.0, 28.0, 10.0, 25.0, 18.0, 31.0, 12.0, 17.0, 23.0, 32.0, 28.0,
12.0, 13.0, 12.0, 22.0, 11.0, 15.0], "episode_types": ["t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t",
"t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t",
"t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t",
"t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t", "t",
"t", "t", "t"]}

{"stats":
"openaigym.episode_batch.2.18416.stats.json"
, "videos": [], "env_info": {"gym_version":
"0.15.3", "env_id": "CartPole-v0"}}

对结果进行分析：测试包括 100 个小片段，即进行一百次测试，测试结果分别为 10，13，25，24，43……即按照该策略操作小车，多数情况下可以坚持 10-30 次操作，结果基本保持稳定。

第一次以熟悉操作流程为主，通过对实例代码的运行和学习，了解在 OpenAI Gym 平台设计算法并进行测试的方法。接下来一段时间将致力于设计算法，编写代码，提高 CartPoleEnv 的得分。并在完成该模型学习之后继续对其他模型及算法的研究。