人工智能导论第一次实验报告

Rule-based Agent of MountainCar in OpenAI Gym

刘雨佳 2017202124

1. OpenAI Gym

（1） 安装：pip install gym

（2） OpenAI Gym 中的智能体可以通过三种基本方法与环境交互。重置 reset：重置环境并返回观测值；执行 step：在环境中执行一个时间步长，并返回观测值 observation、奖励 reward、状态 done 和信息 info；回馈 render：回馈环境的一个帧，比如弹出交互窗口。

（3） 测试：使用 CartPole-v0 的例子进行测试。

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for step in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(step+1))
            break
```

```
[-0.02216001  0.77821409 -0.05124532 -1.22084329]
[-0.00659573  0.58378863 -0.07566219 -0.94464748]
[ 0.00508005  0.77984377 -0.09455513 -1.26011228]
[ 0.02067692  0.58604995 -0.11975738 -0.99847787]
[ 0.03239792  0.7825516  -0.13972694 -1.32624379]
[ 0.04804895  0.58944264 -0.16625181 -1.08034957]
[ 0.05983781  0.39685876 -0.1878588  -0.84410996]
[ 0.06777498  0.20472854 -0.204741   -0.61589124]
Episode finished after 26 timesteps
```

2. MountainCar 环境介绍

MountainCar 属于经典控制问题，目标是在尽可能少的步数内把动力不足的车开到山顶（0.5 位置）。起始在-0.6 到-0.4 的随机位置，速度为 0，当到达目标位置或进行了 200 次时，中止操作。游戏中可以根据观测到的车的位置和速度信息，给出行为决策。每进行一步奖励-1，直到达到中止状态。

观测值：位置和速度

Observation	Min	Max
position	-1.2	0.6
velocity	-0.07	0.07

行为：三个离散值

Num	Action
0	push left
1	no push
2	push right

3. 算法实现

（1） 随机模式

与 CartPole 的测试代码基本相同。由于获取行为的随机性，影响相互抵消，导致小车一直在低处徘徊，不能到达山顶。

```
#random mode
def test(self,steps):
    env.reset()
    total_reward=0
    for s in range(steps):
        env.render()
        action=env.action_space.sample()#get action randomly
        _,reward,done,_=env.step(action)
        total_reward+=reward
        if done:
            break
    return total_reward
```

（2） 特定规则模式

对于这个实验，最关键的地方在于 action 的选择策略。这里的策略是：当小车向左行驶时，若速度不大且位置不高，需要向左推动，这样可以增加速度；当小车即将到达右边山顶但速度很小时，不推动，因为向右推动也可能无法到达山顶反而会反向到左边使得步数增多；一般来说小车向右行驶时都需要向右推动；其余情况不推动。

```
#specific rule mode
def step(self,steps):
    observation=env.reset()
    total_reward=0
    for s in range(steps):
        env.render()#render the env every step
        action=self.get_action(observation)#get action for car
        #0 push left, 1 no push, 2 push right
        observation,reward,done,_=env.step(action)
        total_reward+=reward
        if done:
            print("Episode finished after {} timesteps. Total reward:{}"
                  .format(s+1,total_reward))
            break
    return total_reward
```

```python
def get_action(self,observation):
    pos=observation[0] #position
    vel=observation[1] #velocity
    if pos<self.goal_position/4 and -self.max_speed/2<vel<0:
        return 0
    elif pos>self.goal_position-0.02 and 0<vel<self.max_speed/10:
        return 1
    elif vel>0:
        return 2
    else:
        return 1
```

- 结果：采用此策略，平均 **127** 步可以到达山顶。

```
Average reward of random mode equals to -200.0
Episode finished after 171 timesteps. Total reward:-171.0
Episode finished after 127 timesteps. Total reward:-127.0
Episode finished after 101 timesteps. Total reward:-101.0
Episode finished after 112 timesteps. Total reward:-112.0
Episode finished after 124 timesteps. Total reward:-124.0
Average reward equals to -127.0
```