RL_Gym_Environment_Week_1

October 20, 2020

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. This link contains a standardized set of environments.

This time we will use the "CartPole" as an example.

[1]:
```
import gym
env = gym.make("CartPole-v0")
```

1 Introduction

[3]:
```
env.action_space
```

[3]:
```
Discrete(2)
```

Action of an environment. **Discrete** allows a fixed range of non-negative discrete numbers. In CartPloe, action is either 0 (to the left) or 1 (to the right).

[4]:
```
env.action_space.sample()
```

[4]:
```
1
```

Generate a random action. In CartPloe, it could be 0 or 1.

[5]:
```
env.observation_space
```

[5]:
```
Box(-3.4028234663852886e+38, 3.4028234663852886e+38, (4,), float32)
```

Observation of an environment. In CartPloe, there are four dimensions which are Cart Position, Cart Velocity, Pole Angle, Pole Velocity.

[6]:
```
(env.observation_space.high, env.observation_space.low)
```

[6]:
```
(array([4.8000002e+00, 3.4028235e+38, 4.1887903e-01, 3.4028235e+38],
       dtype=float32),
 array([-4.8000002e+00, -3.4028235e+38, -4.1887903e-01, -3.4028235e+38],
       dtype=float32))
```

Box a multi-dimensional vector of numeric values, the upper and lower bounds of each dimension are defined by Box.low and Box.high.

[7]:
```
env.reset()
```

[7]:
```
array([ 0.03350904, -0.00926525,  0.02891186, -0.03306448])
```

Reset the env to the original setting and return the initial observation.

[9]:
```
env.render(), env.close()
```

[9]:
```
(True, None)
```

Render and close the user interface of the environment.

[11]:
```
state, reward, done, info = env.step(0)
print(state, reward, done, info)
```

```
[  3.71924054   2.73576363 -11.946679    -6.3506755 ] 0.0 True {}
```

Applies one action in the env which should be compatible with env.action_space.

Gets back the new observation state (env.observation_space), a reward (float), a done flag (bool), and other meta information (dict). If done=True, the episode is complete and we should reset the env to restart.

2 Generate a random move loop

[12]:
```
env.reset()
for _ in range(100):
    env.render()
    env.step(env.action_space.sample()) # take a random action
env.close()
```

Apply 100 actions which every action is random (either left or right).

3 Generate a move loop with a simple rule

[5]:
```
class Agent():
    def __init__(self, env):
        self.action_size = env.action_space.n

    def get_action(self, state):
        pole_angle = state[2]
        action = 0 if pole_angle < 0 else 1
        return action

env.reset()
agent = Agent(env)
state = env.reset()
```

```
for _ in range(200):
    env.render()
    action = agent.get_action(state)
    state, reward, done, info = env.step(action) # 4 ob se

    #print(state, reward, done, info)
env.close()
```

```
/Users/yuetongliu/gym/gym/logger.py:30: UserWarning: WARN: You are calling
'step()' even though this environment has already returned done = True. You
should always call 'reset()' once you receive 'done = True' -- any further steps
are undefined behavior.
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
```

Define a function which move the cart with the angle of the pole.

Apply the function in to CartPole enviroment and generate 200 actions.