孙一丹 2016201013

OpenAI Gym 第一阶段报告

1.OpenAI Gym安装

安装

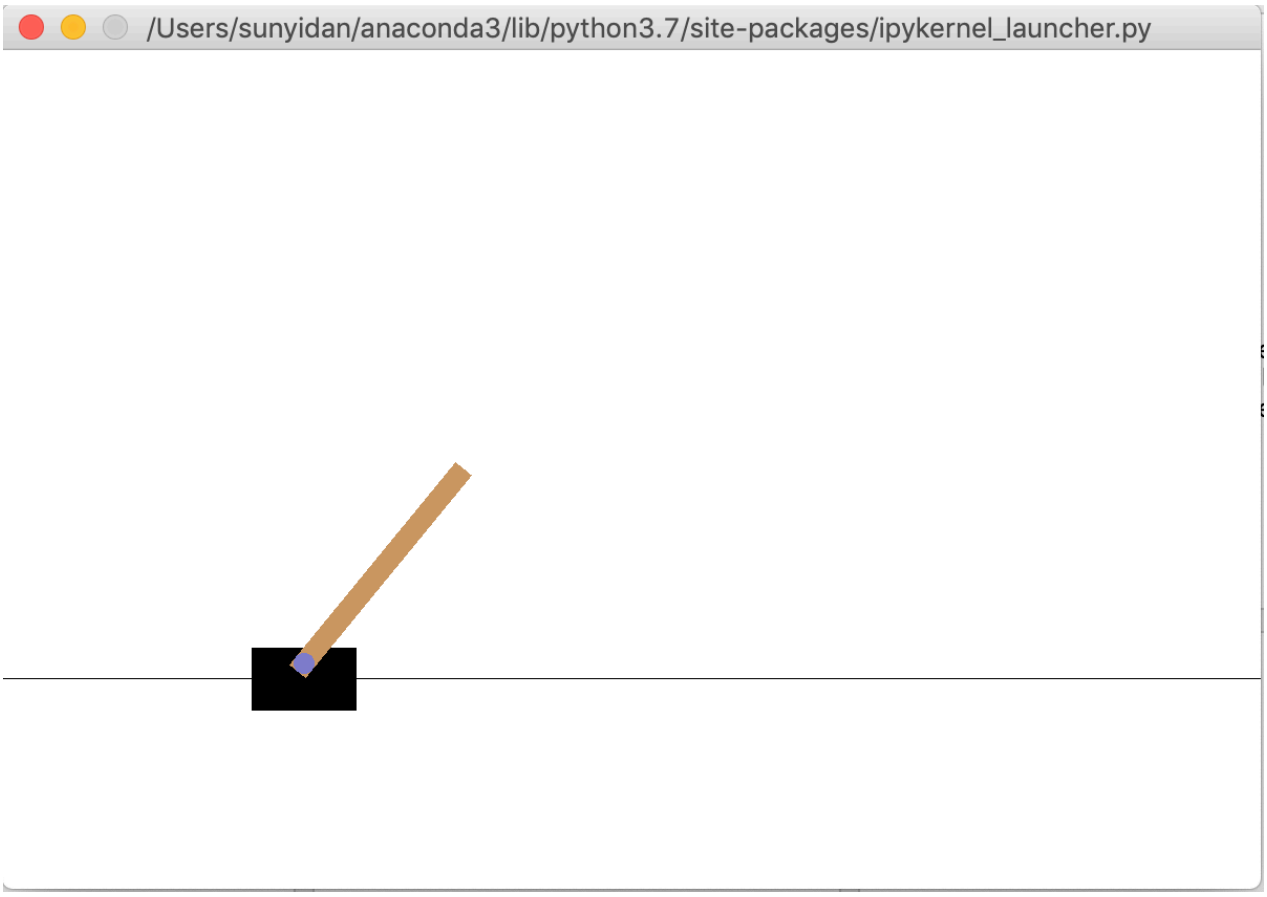
在OS X环境下，打开terminal，使用`pip install gym`安装OpenAI Gym。

测试

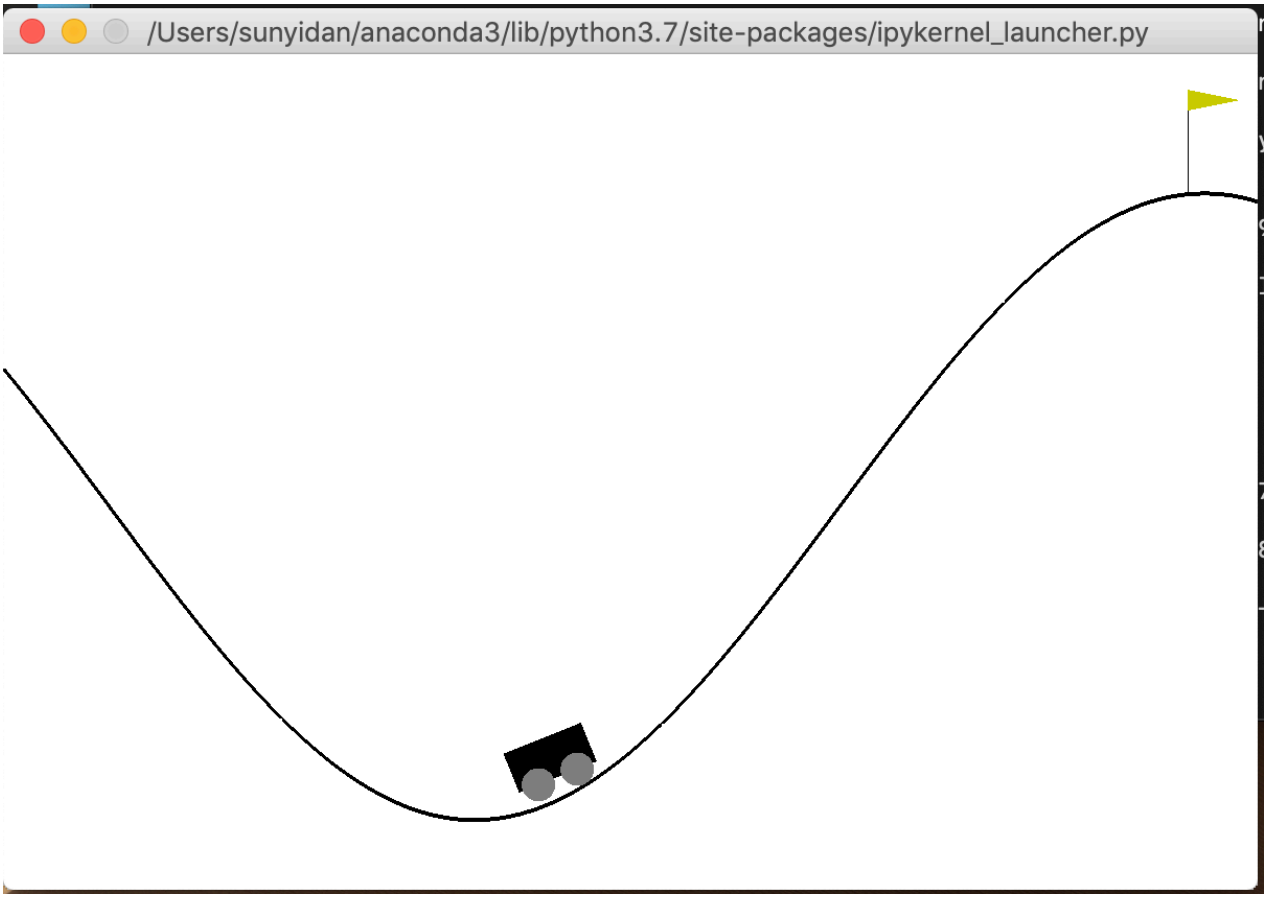
安装后，打开Jupyter Notebook 运行demo以验证是否顺利安装，这里使用Gym官方文档里给出的CartPole-v0下的1000帧验证：

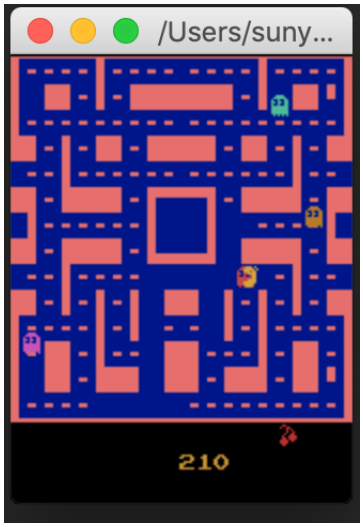
```
import gym #载入
env = gym.make('CartPole-v0') #构建初始环境
env.reset()  #重置环境
for _ in range(1000):  #1000帧
    env.render()  #每一帧重新渲染环境
    env.step(env.action_space.sample()) # take a random action
```

运行结果如下图所示



Gym中提供了多种环境，可以将`'CartPole-v0'`替换为`MountainCar-v0``MsPacman-v0`等，运行结果如下：





可以通过如下代码查看OpenAI Gym中所有环境的列表：

```
from gym import envs
print(envs.registry.all())
```

结果如下：

```
In [19]: print(envs.registry.all())
```

```
dict_values([EnvSpec(Copy-v0), EnvSpec(RepeatCopy-v0), En
vSpec(ReversedAddition-v0), EnvSpec(ReversedAddition3-v0)
, EnvSpec(DuplicatedInput-v0), EnvSpec(Reverse-v0), EnvSp
ec(CartPole-v0), EnvSpec(CartPole-v1), EnvSpec(MountainCa
r-v0), EnvSpec(MountainCarContinuous-v0), EnvSpec(Pendulu
m-v0), EnvSpec(Acrobot-v1), EnvSpec(LunarLander-v2), EnvS
pec(LunarLanderContinuous-v2), EnvSpec(BipedalWalker-v2),
EnvSpec(BipedalWalkerHardcore-v2), EnvSpec(CarRacing-v0),
EnvSpec(Blackjack-v0), EnvSpec(KellyCoinflip-v0), EnvSpec
(KellyCoinflipGeneralized-v0), EnvSpec(FrozenLake-v0), En
vSpec(FrozenLake8x8-v0), EnvSpec(CliffWalking-v0), EnvSpe
c(NChain-v0), EnvSpec(Roulette-v0), EnvSpec(Taxi-v3), Env
Spec(GuessingGame-v0), EnvSpec(HotterColder-v0), EnvSpec(
Reacher-v2), EnvSpec(Pusher-v2), EnvSpec(Thrower-v2), Env
Spec(Striker-v2), EnvSpec(InvertedPendulum-v2), EnvSpec(I
nvertedDoublePendulum-v2), EnvSpec(HalfCheetah-v2), EnvSp
ec(HalfCheetah-v3), EnvSpec(Hopper-v2), EnvSpec(Hopper-v3
), EnvSpec(Swimmer-v2), EnvSpec(Swimmer-v3), EnvSpec(Walk
er2d-v2), EnvSpec(Walker2d-v3), EnvSpec(Ant-v2), EnvSpec(
```

2.OpenAI Gym使用

Observation&reset

前面`CartPole-v0`的例子中，action是随机的，如果想要在每个步骤中做出比采取随机行动更好的action，则需要了解action对环境的影响。

环境的`step`函数返回需要的信息，共有四个值：

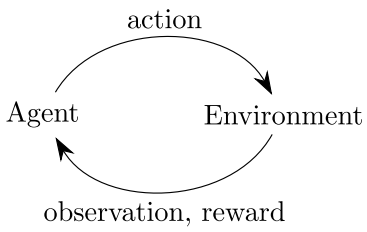
`observation`(**object**):一个与环境相关的对象描述你观察到的环境，如相机的像素信息，机器人的角速度和角加速度，棋盘游戏中的棋盘状态。

`reward`(**float**):先前行为获得的所有回报之和，不同环境的计算方式不一样，但目标总是增加自己的回报。

`done`(**boolean**):判断是否需要重新设定（reset）环境，大多数任务为明确定义的episodes，为`True`时表示episode已终止。

`info`(**dict**):用于调试的诊断信息，有时也用于学习，但正式的评价不允许使用该信息进行学习

在每一个时间步长，agent都会选择一个action，environment将返回一个observation和reward。



如下代码示例为，通过调用`reset()`来启动`CartPole-v0`，返回一个初始`observation`。当`done`返回为`True`时，终止当前的episode：

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t 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(t+1))
            break
```

每次立杆将要倒下去时，环境重置，并且每帧返回`observation`（即立杆位置的观察信息），其中一次episode的观察值如下：

```
[ 0.00448669  0.40505763 -0.05663757 -0.67430987]
[ 0.01258785  0.21076665 -0.07012377 -0.39998326]
[ 0.01680318  0.40680963 -0.07812343 -0.71392451]
[ 0.02493937  0.21285116 -0.09240192 -0.44681913]
[ 0.0291964   0.4091505  -0.10133831 -0.76713953]
[ 0.03737941  0.60551074 -0.1166811  -1.08991129]
[ 0.04948962  0.41210389 -0.13847932 -0.83600009]
[ 0.0577317   0.60881817 -0.15519932 -1.16882972]
[ 0.06990806  0.41601749 -0.17857592 -0.92855033]
[ 0.07822841  0.61304209 -0.19714693 -1.27161023]
Episode finished after14 timesteps
```

Spaces

每个环境都带有描述有效动作和观察结果的space对象：

```
import gym
env = gym.make('CartPole-v0')
print(env.action_space)
#> Discrete(2)
print(env.observation_space)
#> Box(4,)
```

在`CartPole-v0`环境中，action_space为Ddiscrete(2)，即有效的动作为0或1（立杆的运动只有向左和向右）；observation_space为Box(4,)，表示立杆在一个二维空间中，所以有效的观察将是4个数字的数组。也可以通过如下的代码块检查Box的范围：

```
print(env.observation_space.high)
print(env.observation_space.low)
```

结果如下：

```
print(env.observation_space.high)
```

```
[4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]
```

```
print(env.observation_space.low)
```

```
[-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e
+38]
```

3.总结

在第一阶段，我主要进行了OpenAI Gym的安装与配置，并详细阅读了官方文档中对其的介绍，大致了解了`CartPole-v0`环境下各种函数与参数的意义，为下一阶段的项目做好了准备。