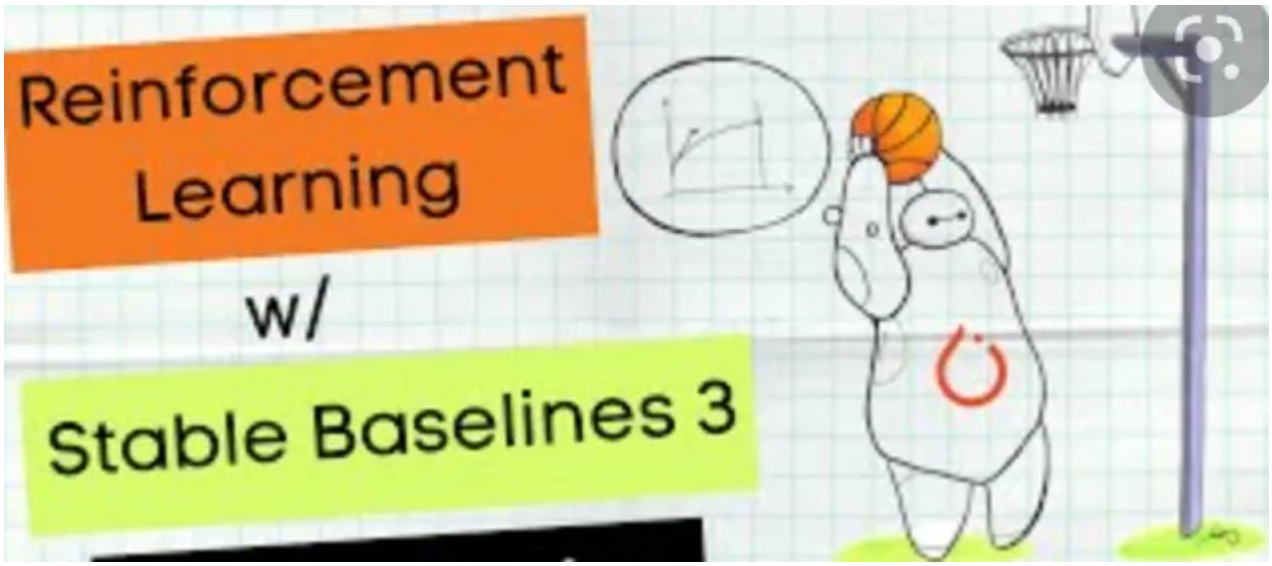
Renee LIN · Follow
4 min read · May 23, 2022






OpenAI has created the Gym and Stable Baselines library to make reinforcement learning easy to use. I'd like to recap how to use it with one of the classic control problems — Mountain Car Continuous. The Colab notebook is here https://colab.research.google.com/drive/1m5Ppsrv6B5maUJ-vMgbZtMeSxqFfUVSP?usp=sharing

## 1. Background

### (1) Deep Reinforcement Learning (RL)

RL was used to train an agent winning the world champion of GO in 2016. Since then, RL has been attracting increasing attention. Different from supervised learning which tries to learn the distribution boundaries or unsupervised learning which learns the distribution directly, RL is created based on Markov Decision Process(**MDP**). MDP is a mathematical formulation of a sequential decision-making process with objectives. In a given **environment**, the learning agent takes **actions** based on its **observation**. The environment will update its state influenced by those actions and give feedback or **reward** back to the agent. The process goes on and on until the agent reaches the goal or meets termination conditions. The learning agent is trying to obtain the **optimal policy**(taking which action under certain observations/states) leading to **accumulated rewards** in one process.

Currently, Proximal Policy Optimization (PPO) is the most used algorithm to solve this MDP problem.

### (2) OpenAI Gym

In RL, the environment is crucial since it provides the reward that the agent's learning is based on. It also needs to update the states in each step. In order to better benchmark the research in various environments and allow people to focus on algorithm development, OpenAI creates a Gym library providing several standard environments. https://www.gymlibrary.ml/

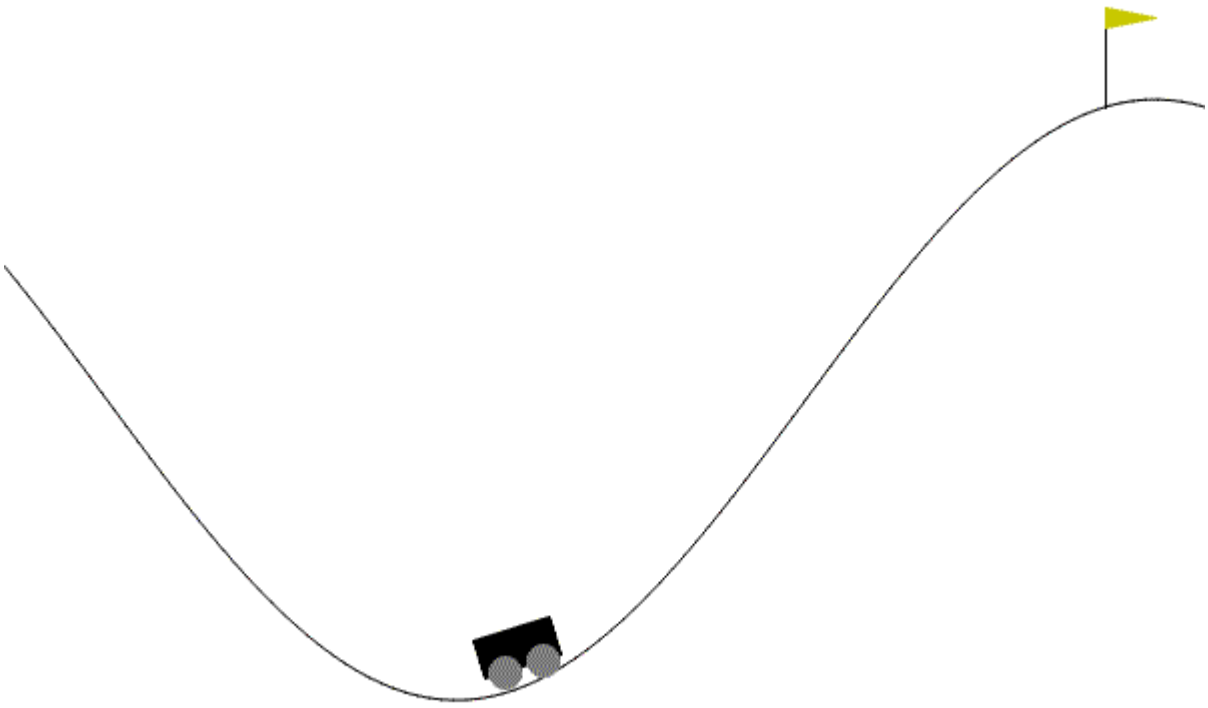
### (3) Stable Baselines3

Stable Baselines3 gives reliable implementations of reinforcement learning algorithms in PyTorch which is the major version replacing previous Stable Baselines. Official Doc https://stable-baselines3.readthedocs.io/en/master/index.html

## 2. Mountain Car Continuous

"The goal of the MDP is to strategically accelerate the car to reach the goal state on top of the right hill. There are two versions of the mountain car domain in gym: one

with discrete actions and one with continuous. This version is the one with continuous actions.”
https://www.gymlibrary.ml/environments/classic_control/mountain_car_continuous/



| | |
|---|---|
| Action Space | Box(-1.0, 1.0, (1,), float32) |
| Observation Shape | (2,) |
| Observation High | [0.6 0.07] |
| Observation Low | [-1.2 -0.07] |
| Import | `gym.make("MountainCarContinuous-v0")` |

Position: -1.2 ~0.6
Speed: -0.07~0.07

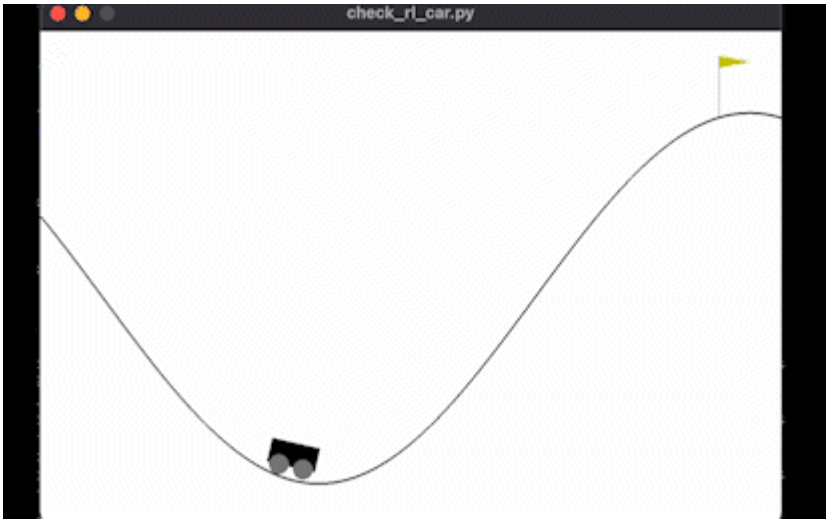
## Reward

A negative reward of *-0.1 * action*$^2$ is received at each timestep to penalise for taking actions of large magnitude. If the mountain car reaches the goal then a positive reward of +100 is added to the negative reward for that timestep.

## 3. Implementation using Stable Baselines3(SB3)

The code is super simple using the library, below is the tuned PPO model which solves the problem. The notebook:https://colab.research.google.com/drive/1m5Ppsrv6B5maUJ-vMgbZtMeSxqFfUVSP?usp=sharing

This is the result I got, it reaches the flag on the right. It tried several times to go to the top.




(1) Install packages

```
pip install stable-baselines3[extra]
import gym

from stable_baselines3 import PPO
from stable_baselines3.ppo import MlpPolicy
from stable_baselines3.common.env_util import make_vec_env

import os
import time
```

(2) Create folders to save models and logs

```
# Saving logs to visulise in Tensorboard, saving models
models_dir = f"models/Mountain-{time.time()}"
logdir = f"logs/Mountain-{time.time()}"
if not os.path.exists(models_dir):
    os.makedirs(models_dir)
```

```
if not os.path.exists(logdir):
    os.makedirs(logdir)
```

(3) Create an Environment

```
# Parallel environments

env = make_vec_env("MountainCarContinuous-v0", n_envs=1)
```

(4) Set up the model with SB3

```
# The learning agent and hyperparameters
model = PPO(
    policy=MlpPolicy,
    env=env,
    seed=0,
    batch_size=256,
    ent_coef=0.00429,
    learning_rate=7.77e-05,
    n_epochs=10,
    n_steps=8,
    gae_lambda=0.9,
    gamma=0.9999,
    clip_range=0.1,
    max_grad_norm =5,
    vf_coef=0.19,
    use_sde=True,
    policy_kwargs=dict(log_std_init=-3.29, ortho_init=False),
    verbose=1,
    tensorboard_log=logdir
    )
```

(5) Training

```
#Training and saving models along the way
TIMESTEPS = 20000
for i in range(10):
    model.learn(total_timesteps=TIMESTEPS,reset_num_timesteps=False,
tb_log_name="PPO")
    model.save(f"{models_dir}/{TIMESTEPS*i}")
```

(6) Load the best model to check the result

```
# Check model performance
# load the best model you observed from tensorboard - the one reach
the goal/ obtaining highest return
models_dir = "models/Mountain-1653282767.3143597"
model_path = f"{models_dir}/80000"
best_model = PPO.load(model_path, env=env)

obs = env.reset()
while True:
    action, _states = best_model.predict(obs)
    obs, rewards, dones, info = env.step(action)
    # env.render()  use Python IDE to check, I havn't figure out how
to render in Notebook
```

A good post about hyperparameters for PPO :

PPO Hyperparameters and Ranges https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe

Reinforcement Learning | Stable Baselines | OpenAI | Openai Gym

Follow

## Written by Renee LIN

1.2K Followers

Passionate about web dev and data analysis. Huge FFXIV fan. Interested in healthcare data now.

### More from Renee LIN