推荐的**RL**入门博客

- 刘建平Pinard

搭建环境

1. 常用**RL**算法库

- tianshou
- rllab
- Deep Reinforcement Learning for Keras
- baselines
- Stable Baseline3（里面有几个模仿学习算法：BC, DAgger, AIRL, GAIL）
- MuJoCo-PyTorch (这个我觉得代码写得非常好，逻辑很清楚，方便修改)

2.常用实验环境

- MuJoCo (最好使用linux环境，然后用学校邮箱去MuJoCo官网申请使用权)
- MiniGrid（最常用的稀疏奖赏环境）
- mj_envs

在本地安装mj_envs环境；

git clone mj_envs

下载mjrl

```
conda activate gymlab
cd ~/下载/mj_envs
pip install -e .
cd ~/下载/mjrl
pip install -e .

cd ~/下载/mj_envs/mj_envs
python utils/visualize_env.py --env_name hammer-v0
```

(1)若出现**GLEW error**

```
Running trained model
Creating window glfw
ERROR: GLEW initalization error: Missing GL version

Press Enter to exit ...Killed
```

是mujoco依赖包没安装完整

```
sudo apt-get install libgl1-mesa-dev libgl1-mesa-glx libglew-dev
libosmesa6-dev build-essential libglfw3
```

然后，参考GLEW error

```
vi ~/.bashrc
export LD_PRELOAD=/usr/lib/x86_64-linux-
gnu/libGLEW.so:/usr/lib/nvidia-390/libGL.so
或者
export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libGLEW.so
sourcec ~/.bashrc
```

(2)No module named 'torch'

```
conda install pytorch==1.4.0 # https://pytorch.org/get-
started/previous-versions/
pip install --upgrade numpy
```

导入环境

```
import mj_envs
import gym

env_name = 'hammer-v0'
env = gym.make(env_name)
```

RL算法tricks

1. 连续动作离散化

经常会遇到一些连续动作空间大的情况，例如某一动作的取值范围为[-100,100]，直接学是很困难的，可以先将其离散化再学习会比较有效。

2. reward shaping

在一些比较复杂的环境中（例如稀疏环境），直接使用环境提供的reward来训练智能体是很困难的，此时可以使用reward shaping来进行初步阶段的预训练。 由于reward shaping容易让学得的目标有偏差，因此可以使用衰减因子来减少这一偏差。

$r_t = a_t s_t + (1 - a_t)[t == T]R$

a_t为线性退火因子。直观来说，做某一个动作对最终目标很有利，那么我们可以人为定义一个奖赏R给智能体，但是为了防止智能体过利用该奖赏，我们可以将其慢慢衰减。

3. 一些好的针对稀疏环境的有效策略

- BEBOLD(一种十分有效的内在奖励定义方式)
- count based (内在奖赏)
- curiosity driven (好奇心驱动)
- UPGO
- Self-Imitation (自模仿)
- UCB采样 (较好的 E&E (Explore and Exploit) 方法)
- 汤普森采样 (UCB的贝叶斯版)

4. self-play 技巧

- 多样性：为防止最终“左右互搏”学得的策略过于单一，可以在self-play过程中随机采样历史模型来对抗
- 分步优化：为防止两个智能体落入能力低下的纳什均衡解，可以分开来优化。例如，训练下围棋的智能体，可以先防守方，只训练进攻方，当胜率打到某个值时（55%）；再固定进攻方，只训练防守方。

5.处理GAN训练不稳定性参考博客

（1）加噪：给expert_data, generate_data加噪然后再输入discriminator训练；

（2）标签平滑：将expert_data的label由1->0.9，generate_data的label由0->0.1；

```
# generator_loss =
tf.nn.sigmoid_cross_entropy_with_logits(logits=generator_logits,
labels=tf.zeros_like(generator_logits))
generator_loss =
tf.nn.sigmoid_cross_entropy_with_logits(logits=generator_logits,
labels=tf.add(tf.zeros_like(generator_logits), 0.1))
# expert_loss =
tf.nn.sigmoid_cross_entropy_with_logits(logits=expert_logits,
labels=tf.ones_like(expert_logits))
expert_loss =
tf.nn.sigmoid_cross_entropy_with_logits(logits=expert_logits,
labels=tf.add(tf.ones_like(expert_logits),-0.1))
```

（3）调整generator和discriminator学习率：

如调整为：d_lr（0.0004） > g_lr（0.0001）；

原来：d_lr(3e-4=0.0003) < g_lr(1e-3=0.001)

6.结合课程学习进行轨迹增广

尝试用增广轨迹长度的倒数作为权重，head->middle->tail有序利用增广后的轨迹进行模仿学习。