强化学习：作业五

丁云翔 191250026

2022 年 1 月 24 日

1 作业内容

在 d4rl 的 Hopper 数据集上应用离线强化学习算法 CQL。

2 实现过程

使用了 CQL 的官方实现，由于其官方实现已经支持了运行 d4rl 的 Hopper 数据集，只需在运行的参数中指定具体环境即可。CQL 算法的具体实现位于 rlkit/torch/sac/cql.py。

3 复现方式

环境要求同 CQL 官方实现的要求，需要安装 rlkit 和 d4rl，同时 torch 版本不低于 1.1.0。
对于任务 hopper-random，在 code 文件夹下，运行

```
python examples/cql_mujoco_new.py
--env=d4rl-mujoco-hopper-random-v2
--policy_lr=1e-4 --seed=10 --lagrange_thresh=-1.0
--min_q_weight=5.0 --gpu=0
--min_q_version=3
```

对于任务 hopper-medium，在 code 文件夹下，运行

```
python examples/cql_mujoco_new.py
--env=d4rl-mujoco-hopper-medium-v2
--policy_lr=1e-4 --seed=10 --lagrange_thresh=-1.0
--min_q_weight=5.0 --gpu=0
--min_q_version=3
```

对于任务 hopper-expert，在 code 文件夹下，运行

```
python examples/cql_mujoco_new.py
--env=d4rl-mujoco-hopper-expert-v2
--policy_lr=1e-4 --seed=10 --lagrange_thresh=-1.0
--min_q_weight=5.0 --gpu=0
--min_q_version=3
```

4 实验效果

因为我的笔记本的 gpu 不支持 cuda，用 cpu 跑很长时间也跑不出结果，所以暂无实验效果。

5 思考

关于值外推问题，因为 model-based 算法对环境建模，学到了状态转移模型，所以泛化性要强于 Q-function，一定程度上覆盖到了优化策略采样的轨迹终端。另外值外推问题出现的原因是无法对数据集外的状态-动作对进行采样，而 model-based 算法由于学到了状态转移模型，这一问题得以缓解。

关于算法何时停止，可以根据数据集规模、问题的复杂程度和以往经验得到一个大致的迭代轮数，并以此作为限制。如果是 model-based 的方法，也可以使用学得的状态转移模型对策略进行测试。

6 小结

离线强化学习算法比前几次实验的强化学习算法还是要更复杂许多的，且由于依然需要使用 pytorch，在自己的笔记本上还是跑不出结果，不太好去自己实现和测试，正好 CQL 算法的官方实现支持 d4rl 数据集，所以就直接使用了官方的实现。