强化学习-2022秋-课程作业五

作业内容

探究离线强化学习值外推误差的问题，并在d4rl的Hopper数据集上训练离线强化学习算法，并思考如何评估策略性能以及判断算法的收敛。

作业描述

数据与环境描述

本次作业的数据集为d4rl的Hopper数据集，共有三个任务：hopper-random，hopper-medium，hopper-expert。random为随机策略在环境中采样获得的数据集，medium为策略学习到中等性能时在环境中采样获得的数据集，expert为专家策略在环境采样获得的数据集。

本次作业在课程网站中提供了数据集的下载，也可以从GitHub开源的d4rl获得。关于d4rl，详情可见[1]。

关于实验环境，数据集对应的环境为gym mujoco的Hopper-v2。需要注意的是，本次作业并不需要和环境交互。也并不建议使用环境直接评估学习到的策略，关于策略的评估，见提交方式。

关于作业提供的数据集的读取：

```
import numpy as np

task_name = "{}-{}-v0"
env_names = ['halfcheetah', 'hopper', 'walker2d']
levels = ['random', 'medium', 'expert']

def load_data(path):
    data = np.load(path, allow_pickle=True).item()
    states = data['state']
    actions = data['action']
    next_states = data['next_state']
    rewards = data['reward']
    terminals = data['terminal']

    dataset = {'state': states,
               'action': actions,
               'next_state': next_states,
               'reward': rewards,
               'terminal': terminals}

    return dataset
```

```
for env_name in env_names:
    for level in levels:
        dataset = load_data('./dataset_mujoco/{}_{}_data.npy'.format(env_name,
level))
```

** 关于Hopper任务： **

状态空间：11维，各维度取值范围连续。

动作空间：3维，各维度取值范围(-1,1)，连续。

奖励函数：奖励函数为x方向上前进的长度、对动作幅度的惩罚（减去动作各维度的平方和）以及是否存活的奖励（如果当前步没有倒下则+1）。

转移函数：几乎是确定性的转移。

最大步长：1000。

问题与任务描述

首先我们举个例子来简要说明值外推带来的问题。在之前的作业中，我们尝试了Q-table求解强化学习。那么现在我们暂时不考虑Q-value的泛化能力，先思考如下的迷宫问题，在该例子中，黄色为出发点，棕色为墙壁，粉色为可通行区域，红色为离线数据集中的数据。每个格子都有两个数值，第一个数值为单步的奖励，第二个数值为我们的Q-table各个动作下初始化的数值。

-5，x	0，x	-2，x	0，x	0，x	-1，x	0，x	0，x	0，x	7，x
0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	5，x
0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x
0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	-4，x
0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x
0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	0，x	-2，x

请思考：

2. 当x=-1时，我们直接在离线数据集上进行动态规划，Q-value会发生什么变化？我们能否获得一个可以走到最大奖励的策略？
3. 那么当x=20呢？（认为折扣因子为1）。

该例子较为简单，也没有考虑引入值函数近似后的泛化性，并不能说明离线强化学习全部的问题。但是基于该例子，我们可以对离线强化学习值外推误差有一个初步的认识：由于无法对数据集外的状态-动作对进行采样，在进行策略迭代时，如果需要使用数据集外的状态-动作值计算Q-target，来更新当前的Q值时，一旦该外推的值过大，将可能导致整个数据集上的Q值估计错误，从而导致导出的策略完全失败。

以往的工作对值外推的解决可以简单分类如下：

2. 集成多个Q-function，然后计算Q-target时，取它们的最小值。
3. 不使用数据集外的状态-动作对更新Q-target，例如将优化策略限制在数据集采样策略的近邻、降低数据集外的Q值等等。
4. Model-based方法。（请思考，为什么当前的model-based算法可以一定程度上缓解该问题？提供一个思考角度：学到的模型的泛化性要强于Q-function，一定程度上覆盖到了优化策略采样的轨迹终端）

请完成：

2. 思考探究值外推问题。
3. 实现任意一种已有的离线强化学习算法（如BCQ[2][源代码](https://github.com/sfujim/BCQ)、CQL[3][源代码](https://github.com/aviralkumar2907/CQL)、MOPO[4][源代码](https://github.com/tianheyu927/mopo)、BRAC[5]、BEAR[6]、MORel等[7]），并在给定的数据集上训练并导出模型。
4. 在训练过程中，由于没有真实环境可供测试策略，尝试思考算法何时应该停止（不一定是收敛）。
5. ** （选做） ** 尝试评估训练时的策略。
6. ** （选做） ** 尝试改进上述的离线强化学习算法。

代码描述

** 允许自己实现算法，可以不使用本次实验提供的代码框架。 **

实验提供的代码文件夹code由'algorithm_offline'、'dataset_mujoco'、'test_data.py'等文件组成。

'algorithm_offline'：提供了基本的算法框架，model中放置了几个基本的算法，可以基于model中的算法继续实现。** 请在agent文件夹下实现自己的算法。 **

'dataset_mujoco'：存放数据，请将下载好的数据放入该文件夹。

'test_data.py'：展示如何使用算法框架和数据，样例使用的算法是TD3BC[8]。

提交方式

完成的作业请通过sftp上传提交。上传的格式为一份压缩文件，命名为'学号+姓名'的格式，例如'MG21370001张三.zip'。文件中除原有代码外，还需包含实现的算法py文件、'model_random.pt'、'model_medium.pt'、'model_expert.pt'（三个模型文件）和'Document.pdf'（一份pdf格式的说明文档），文档内容至少需要包含：

1. 算法的实现说明（如果实现了一种变体，请额外说明）。
2. 如果有相关的改进，也请在其中说明。
3. 我们会对模型性能进行测评，** 该结果并不是成绩的主要决定因素。 **

文档模板参见'Document5.tex'和'Document5.pdf'。(也可以使用自己的模板。)

参考文献

[1] Justin Fu et al. D4RL: Datasets for Deep Data-Driven Reinforcement Learning.

[2] Scott Fujimoto et al. Off-Policy Deep Reinforcement Learning without Exploration.

[3] Aviral Kumar et al. Conservative Q-Learning for Offline Reinforcement Learning.

[4] Tianhe Yu et al. MOPO: Model-based Offline Policy Optimization.

[5] Yifan Wu et al. Behavior Regularized Offline Reinforcement Learning.

[6] Aviral Kumar et al. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.

[7] Sergey Levine et al. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.

[8] Scott Fujimoto et al. A Minimalist Approach to Offline Reinforcement Learning.