《机器学习》

课程作业-强化学习

1. 作业目标：

（1）完成 OpenAI Gym 环境的安装，并熟悉其中 Classical Control 系列任务中 Mountain Car 任务的状态、动作、回报函数设计；

（2）基于附件 HW3_RL，参考强化学习章节知识，并查阅相关文献材料，完善附件中 DQN（Deep Q Network）代码实现，复现该算法在 Mountain Car 下的实验结果；

（3）提交技术报告，阐述对于 Mountain Car 环境、DQN 代码的个人理解；同时报告里汇报对于 DQN 代码中缺失部分的补全方案以及 Mountain Car 环境下的实验结果；

2. 环境、代码安装

（1）环境安装

参考环境文档：https://www.gymlibrary.dev/，完成 gym 环境安装。

（2）代码安装

附件 HW3_RL 提供 DQN（Deep Q Network）和 AC（Actor Critic）代码框架实现（基于 Python 和 Pytorch），本作业仅考虑 DQN 算法，AC 算法感兴趣的同学可以自行学习。其中，DQN 算法要求同学们补充其中缺失部分方可运行成功。

3. 代码结构简要阐述

整体代码框架由 Agent, algorithm, common, log, model_log, network, runner 和 runs 共 8 个文件夹组成，此外还包括运行函数 main.py。

HW3_RL								
Agent	algorithm	common	log	model_log	network	runner	runs	main

Agent 文件夹下 dqn_agent.py 文件，用于实例化 DQN 智能体类，以及每一时间步下根据状态选择动作的 select_action()函数；

algorithm 文件夹下 dqn.py 文件定义 DQN 算法的核心组件；

common 文件夹下定义有参数设置文件 argument.py，数据存储池文件 replay_buffer.py 以及工具类 utils.py；

log 文件夹用于实时记录 DQN 算法在相应环境下的实验结果；

model_log 文件夹用于存储 DQN 算法周期性的模型参数；

network 文件夹下提供基本的模型结构设计；

runner 文件夹下 runner_dqn.py 用于实现基于 DQN 算法的智能体 Agent 与环境多步交互的过程；

runs 文件夹用于存储 DQN 算法学习过程中的一些记录项，基于 tensorboard 实现；

4. DQN 算法伪代码

Initialize replay buffer $\boldsymbol{D}$
Initialize the state-action value function $\boldsymbol{Q}$ with weights $\boldsymbol{\theta}$
Initialize the target state-action value function $\widehat{\boldsymbol{Q}}$ with weights $\boldsymbol{\theta^-}$
For episode = 1 to M, do:
Reset the environment and receive the initial state $\boldsymbol{s_0}$
Select the action $\boldsymbol{a_0}$ according to the **epsilon-greedy policy**
Execute action $\boldsymbol{a_0}$ in the simulator
Environment transits to the next state $\boldsymbol{s_1}$, returns the reward $\boldsymbol{r_0}$, the terminate flag $\mathbf{done}$
Store the transition $(\boldsymbol{s_0}, \boldsymbol{a_0}, \boldsymbol{r_0}, \mathbf{done}, \boldsymbol{s_1})$ into $\boldsymbol{D}$
If train:
Sample random mini-batch $\{(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \mathbf{done}, \boldsymbol{s_{t+1}}), \dots\}$ from $\boldsymbol{D}$
If done, set: $$\boldsymbol{y = r_{t+1}}$$ otherwise: $$\boldsymbol{y = r_{t+1} + \gamma \max_{a_{t+1}} \widehat{Q}(s_{t+1}, a_{t+1}; \theta^-)}$$
Perform the gradient descent with loss function $(\boldsymbol{y - Q(s_t, a_t; \theta)})^2$ on $\boldsymbol{\theta}$
Set $\boldsymbol{\theta^- = \theta}$ every $\boldsymbol{C}$ steps
End
End

算法详细描述见课程章节 PPT 中，也可阅读课外文献资料理解。

5. DQN 代码实现

附件中 DQN 代码目前无法运行，需要补全 dqn_agent.py 文件中的 epsilon_greedy 相关代码实现，以及 dqn.py 文件中 target_q 的相关代码实现。这两部分对应于上述伪代码中的标

红内容，在代码中搜索 TODO 即可找到。

代码默认参数已经设置完成，如果同学们完成 epsilon_greedy 以及 target_q 相关部分代码实现，则可以设置 argument.py 中的--evaluate=False, --load_model=False 进行训练，该实验基于 CPU 可以较快完成。完成后可切换--evaluate=True, --load_model=True，将 runner.py 中第 102 行 self.env.render()解除注释，加载训练的模型的模型即可获得如附件演示视频所示效果。