Homework

值迭代和策略迭代

在`Frozen Lake`环境中实现值迭代和策略迭代算法。环境及代码模板已在`VI_and_PI`文件夹中提供。

1. **(策略迭代)** 根据`vi_and_pi.py`中提供的代码文件实现`policy_evaluation`函数,`policy_improvement`函数and`policy_iteration`函数. 其中迭代停止的阈值 (即$max_s|V_{old}(s) - V_{new}(s)|$) 被给定为 tol = 10^{-3}，折扣因子 $\gamma = 0.9$。
2. **(值迭代)** 根据`vi_and_pi.py`中提供的代码文件实现`value_iteration`.其中迭代停止的阈值被给定为 tol = 10^{-3}，折扣因子 $\gamma = 0.9$。

Sarsa和Q-learning

在`Maze`环境中实现Sarsa和Q-learning算法。环境及代码模板已在`Sarsa_and_QLearning`文件夹中提供。在迷宫环境中，一个红色的智能体被初试化在一个大小为 4×4的网格迷宫环境中。智能体仅仅只能观测到自己的位置信息。在每个时间步，智能体可以向上下左右四个方向移动。当智能体到达黄色网格处时获得+1的奖励值，到达黑色网格时获得-1的奖励值，其他情况下均不会获得奖励值。

1. **(Sarsa)** 在`RL_sarsa.py`中实现Sarsa算法。
2. **(Q-learning)** 在`RL_q_learning.py`中实现**Q_learning**算法。

DQN

在Atari中实现最基本的DQN算法。环境及代码模板已在`DQN_and_PG`文件夹中提供。Atari环境会返回大小为$(height \times width \times channels)$图片作为智能体的观测。而强化学习的观测一般是采用一维向量的形式，通常只需要利用全连接神经网络，算法就可以收敛。由于Atari环境返回的是图片，因此需要用CNN对图像信息预处理，并将处理完的信息reshape成一维向量的形式作为强化学习神经网络的输入。强化学习算法的评判标准主要有reward收敛值大小、reward收敛速度等。

1. **(DQN)** 在Atari `PongNoFrameskip-v4`环境中实现DQN、Double DQN、Dueling DQN算法。环境最终的reward至少收敛至17.0。

Policy Gradient

在`cartpole`中实现policy gradient及其改进算法。环境及代码模板已在`DQN_and_PG`文件夹中提供。策略梯度更新公式如下：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log \pi_\theta(a|s) Q^{\pi_\theta(s,a)}]$$

1. **(REINFORCE)** 在`cartpole`环境中实现基本的REINFORCE算法，即使用蒙特卡洛采样G_t作为$Q^{\pi_\theta(s,a)}$的无偏估计，更新公式如下：

$$J(\theta) = \frac{1}{\sum T_i} \sum_{i=1}^{|D|} \sum_{t=1}^{T_i} [log \pi_\theta(a_t^i|s_t^i) G_t^i],$$

其中，$\tau^i = (s_0^i, a_0^i, r_0^i, s_1^i, \ldots, s_{T_i}^i, a_{T_i}^i, r_{T_i}^i)$，$D$是在环境中执行策略$\pi_\theta$产生的所有轨迹的集合。最终算法性能的评判标准：环境最终的reward至少收敛至180.0。

2. **(A2C)** 在`CartPole`环境中实现A2C算法.最终算法性能的评判标准：环境最终的reward至少收敛至180.0。

开悟

环境要求

```
tensorflow 1.x cpu 版
numpy
```

作业目录结构

1. `kaiwu`目录为作业的开发目录，主要实现`exp_1.py`和`exp_2.py`的`_build_output_tensors`方法
2. 每个作业提供了`frozen.pb`文件，可通过`netron`可视化完整的网络结构以及各个op与tensor的名字
3. `exp_1.py`为作业1的代码文件，`exp_2.py`为作业2的代码文件

作业

1. 通过`inference.py`生成`exp_1`和`expe_2`目录下的`checkpoint`和`frozen.pb`

   初始化`checkpoint`和`frozen.pb`，执行以下命令之后将会生成`exp_1`和`exp_2`目录以及相关文件

   ```
   python inference.py init exp_1
   python inference.py init exp_2
   ```

2. 作业

   1. 在`exp_1.py`中使用 tensorflow 实现 loss 函数
   2. 在`exp_2.py`中实现给定的网络结构, 并成功加载 checkpoint 进行推理

3. 验证作业结果

   通过`main.py`来验证`exp_1`和`exp_2`的结果是否正确

   - 从代码文件加载测试的网络，从`frozen.pb`加载baseline
   - 随机生成输入数据，分别使用测试网络和baseline得到运行结果，对比两者结果是否一致
   - 重复10次
   - 测试命令

   ```
   python main.py test exp_1
   python main.py test exp_2
   ```

4. 作业开发整体流程

   - 每个作业都提供了`frozen.pb`文件, 该文件提供了整个网络所有 op 和 tensor 的描述,可以通过`netron`进行可视化查看

- 按照 `frozen.pb` 的描述要求, 实现对应的网络代码 (相关名字定义要一致, 否则无法加载 checkpoint)
- 使用 `main.py` 进行测试

PS：

1. 不要求提交作业
2. 相关代码下载地址： https://github.com/ZYC9894/2022AI_Practice/tree/main/Homework
3. 相关环境的说明文档：https://www.gymlibrary.ml/
4. 参考答案：https://github.com/ZYC9894/2022AI_Practice/tree/main/Answer