AI Coding Homework 3: RL

欢迎访问本项目github仓库：AI Coding Homework 3: RL

Problem Formulation

- State Space

$$\mathcal{S} = \{(x, y) | x, y \in \{1, 2, 3, 4, 5, 6\}\}$$

- Action Space

$$\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$$

- Transition Probability

$$P(s'|s, a) = \begin{cases} 1 & \text{if } s' = \text{next}(s, a) \\ 0 & \text{otherwise} \end{cases}$$

- Reward Function

$$r(s, a) = \begin{cases} 1 & \text{if } \text{next}(s, a) == (5, 5) \\ -1 & \text{if } \text{next}(s, a) \in \text{coord}_{stone} \\ 0 & \text{otherwise} \end{cases}$$

Where $\text{next}(s, a)$ is the next state after taking action a from state s, when done is True, the transition ends, next function gives nothing. And coord_{stone} is the set of coordinates of the stones.

Method

本次作业使用的方法有如下的几种，不过由于随机的特性，只有Monte Carlo Control和Q learning我在某几次的实验中跑出了收敛的结果，但是大部分时候经常会卡住，所以我加入一个叫Hot_start的方法，开始现在目标附近训练，然后再全图随机训练; 然后Value Iteration的效果是最好的，每次都会收敛。超参数详见默认的超参数，$\gamma = 0.9$时效果最棒。

- **Value Iteration**: Super effective
- **Monte Carlo Control**: Super slow because it is easy to get stuck by the stones
- **Q learning**: Super slow, also by the stones

代码说明

- `agent.py` 中包含了Agent类，这个类中包含Value Iteration; AgentMC类中包含了Monte Carlo Control和Q learning的方法。
- `maze_env.py` 中包含了环境的定义，以及训练和测试的代码。在其中我新加了random_yoki, set_yoki的方法，用于随机生成和设置yoki的位置。然后也设置了args，用于接收命令行参数。更多模式有random_walk, og(on_given)用于测试Agent.

Run

更多参数的使用详见代码，vi, og, rw模式不需要设置超参数，因为参数已经默认设计完成，og模式需要在agent.py的对应函数设置好轨迹。

```
# train the agent
python maze_env.py --mode train --algo vi # Value Iteration, mc, q for Monte Carlo Control, Q le

# test the trained policy, only for mc and q
python maze_env.py --mode test --ckpt agent.pkl
```