RL组

强化学习（RL）是机器学习（ML）三个组成部分之一，主要研究交互过程中的时序决策问题，目前部分AI顶会有超过五分之一的工作是与RL有关的

在本次任务中，以下两个题目可以被选择，你只需要尽可能达到任务要求。学习过程中希望你能够详细记录学习成果，能够对算法有整体认识，并能够掌握一些RL算法的代码构成并结合具体任务环境使用训练策略

注：不反对使用AI，但在答辩时会问及算法的详细理解，特别是AI成分明显的部分。

多臂老虎机

```
import numpy as np
import matplotlib.pyplot as plt

class BernoulliBandit:
    """
    伯努利多臂老虎机

    参数:
        K (int): 老虎机的臂(拉杆)数量
        seed (int, optional): 随机种子，用于结果复现

    属性:
        probs (np.ndarray): 每个臂的奖励概率(0-1之间的值)
        best_idx (int): 最优臂的索引
        best_prob (float): 最优臂的奖励概率
        K (int): 臂的数量
        counts (np.ndarray): 每个臂被选择的次数
        values (np.ndarray): 每个臂的平均奖励值
        cumulative_regret (float): 累积遗憾
        cumulative_reward (float): 累积奖励
        history (list): 历史选择记录(臂索引, 奖励)
    """

    def __init__(self, K, seed=None):
        """
        初始化伯努利多臂老虎机

        参数:
            K (int): 臂的数量
            seed (int, optional): 随机种子
        """
        if seed is not None:
            np.random.seed(seed)

        self.probs = np.random.uniform(size=K)        # 随机生成K个0~1的数，作为拉动每根拉杆的获得奖
        self.best_idx = np.argmax(self.probs)         # 获奖概率最大的拉杆索引
        self.best_prob = self.probs[self.best_idx]    # 最大的奖励概率
        self.K = K

        # 初始化统计信息
        self.counts = np.zeros(K, dtype=int)          # 每个臂被选择的次数
```

```
        self.values = np.zeros(K)                    # 每个臂的平均奖励值
        self.cumulative_regret = 0.0                 # 累积遗憾
        self.cumulative_reward = 0.0                 # 累积奖励
        self.history = []                            # 历史选择记录(臂索引, 奖励)

    def step(self, k):
        """
        选择指定的臂并返回奖励

        参数:
            k (int): 选择的臂索引(0到K-1)

        返回:
            reward (int): 奖励值(0或1)
        """
        # 检查输入有效性
        if k < 0 or k >= self.K:
            raise ValueError(f"臂索引{k}超出范围(0-{self.K-1})")

        # 生成奖励(伯努利试验)
        reward = 1 if np.random.rand() < self.probs[k] else 0

        # 更新统计信息
        self.counts[k] += 1
        # 使用增量公式更新平均值: new_avg = old_avg + (reward - old_avg) / count
        self.values[k] += (reward - self.values[k]) / self.counts[k]

        # 更新累积奖励和遗憾
        self.cumulative_reward += reward
        self.cumulative_regret += self.best_prob - reward

        # 记录历史
        self.history.append((k, reward))

        return reward

    def reset_stats(self):
        """重置所有统计信息(保持概率分布不变)"""
        self.counts = np.zeros(self.K, dtype=int)
        self.values = np.zeros(self.K)
        self.cumulative_regret = 0.0
        self.cumulative_reward = 0.0
        self.history = []
```

```python
def plot_probabilities(self):
    """可视化每个臂的奖励概率"""
    plt.figure(figsize=(10, 6))
    plt.bar(range(self.K), self.probs, color='skyblue')
    plt.axhline(y=self.best_prob, color='r', linestyle='--', label=f'最优概率: {self.best_pr
    plt.xlabel('臂索引')
    plt.ylabel('奖励概率')
    plt.title('多臂老虎机奖励概率分布')
    plt.legend()
    plt.grid(True, axis='y', alpha=0.3)
    plt.show()

def plot_performance(self):
    """可视化老虎机的性能指标"""
    if not self.history:
        print("没有历史数据可供绘图")
        return

    # 准备数据
    steps = np.arange(1, len(self.history) + 1)
    rewards = np.array([r for _, r in self.history])
    cumulative_rewards = np.cumsum(rewards)
    cumulative_regrets = np.cumsum([self.best_prob - r for r in rewards])

    # 创建图表
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10), sharex=True)

    # 累积奖励图
    ax1.plot(steps, cumulative_rewards, label='累积奖励')
    ax1.plot(steps, self.best_prob * steps, 'r--', label='理论最大奖励')
    ax1.set_title('累积奖励随时间变化')
    ax1.set_ylabel('累积奖励')
    ax1.legend()
    ax1.grid(True)

    # 累积遗憾图
    ax2.plot(steps, cumulative_regrets, label='累积遗憾')
    ax2.set_title('累积遗憾随时间变化')
    ax2.set_xlabel('步数')
    ax2.set_ylabel('累积遗憾')
    ax2.legend()
    ax2.grid(True)
```

```
        plt.tight_layout()
        plt.show()

    def __str__(self):
        """返回老虎机的字符串表示"""
        return (f"BernoulliBandit(K={self.K}, best_arm={self.best_idx}, "
                f"best_prob={self.best_prob:.4f})")

# 测试代码
if __name__ == "__main__":
    np.random.seed(114514)
    K = 10
    bandit = BernoulliBandit(K)
    print("随机生成了一个10臂伯努利老虎机")
    print(f"获得概率最大的拉杆为{bandit.best_idx}号，其获奖概率为{bandit.best_prob:.4f}")

    # 可视化概率分布
    bandit.plot_probabilities()

    # 模拟随机选择策略，用于了解环境运行，交互次数为200次，累计奖励
    print("\n模拟随机选择策略...")
    for _ in range(200):
        arm = np.random.randint(K)  # 随机选择一个臂
        reward = bandit.step(arm)

    #====================================#
    #你的代码，需将前面的随机策略注释

    #====================================#



    # 输出统计信息
    print(f"\n累积奖励: {bandit.cumulative_reward}")
    print(f"累积遗憾: {bandit.cumulative_regret:.2f}")
    print(f"每个臂的选择次数: {bandit.counts}")
    print(f"每个臂的平均奖励: {np.round(bandit.values, 4)}")

    # 可视化性能
    bandit.plot_performance()
```

你被要求：

- 尝试使用你所学到的知识，如动态规划等，尽可能的在多臂老虎机中得到更高的分数（200次累计交互）
- 学习贝尔曼方程，使用贝尔曼方程解决上述问题。

Gym-Taxi

Taxi来自OpenAI提供的开源环境Gym，它在Gym说明文档中被详细介绍了

你被要求：

- 分别使用Sarsa和Q-learning，在Taxi环境中训练和测试，记录测试结果并可视化
- 试图使Agent收敛（在每一个episode中，可以稳定地将乘客送到目的地），如果不能得到收敛的结果，分析原因并继续尝试，记录算法的改进过程
- 解释online和offline的区别。
- 关注你使用的算法中的细节（例如动作的探索（Exploration）和利用（Exploitation）、经验回放（Experience Replay）），分析它们在该算法中被使用的原因
- 了解Reward Shaping技巧，尝试在训练过程中使用它，并记录它带来的影响

另外，Taxi环境的动作和状态都是尺度相当小的离散值（向量），如果你在此环境取得了成功，选择更复杂的环境（例如Atari系列，它们的Observation往往是RGB图像），你可以使用任何技巧和算法，试图在新的环境中得到收敛结果

注：这个环境比较简单，此外可以适当尝试一些atari游戏环境，可以使用各种策略和技巧加速模型收敛

论文阅读

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

DPO是现代LLM RLHF RL-post_training的基石，从以往的基于奖励转为优化概率，阅读这一篇文章，回答以下问题：

- 从前面的问题中，我们知道，RL的loss本质上就是最大化奖励，即min（-reward），那么，本论文中的Loss函数是如何由BT（Bradley-Terry）推导来的？优化奖励是如何转化为优化概率的？
- Eq3 中加入了KL散度，解释为什么需要加入KL散度？（如果你有兴趣，可以阅读PPO的论文，详细探索KL散度在RL中的作用）
- 简要说说你认为RL在LLM上可以有什么应用？有人说，RL的本质是去学习Reward Model，基于本文，你觉得RL是否可以学习到Human的偏好Reward？RL是否可以不需要预训练过程完成对LLM的训练，为什么？
- 阅读实验部分，模型经过这种后训练后，对于未经偏好训练的数据输入表现如何？
- （可选）尝试使用DPO解决之前提到的两个问题

Reinforcement Pre-Training

这一篇文章是第一个RL预训练模型， 是RL从后训练角色迈向预训练的一大步，回答以下问题

- 模型的训练数据/语料是什么格式?
- 模型的奖励是如何定义?
- 作者使用了哪些RL方法进行pretrain？了解这些方法并解释其在RLpretrain中的作用和必要性。

自定义

你可以在以下任务中选择一项：

- 建立一个连续动作空间的仿真环境（或使用一个感兴趣的），在该环境中使用至少一种连续环境下的RL算法得到结果并展示
- 选择一个RL前沿方向深入研究（如多智能体，LLM post trainning，具身智能 etc...），你只需要详尽地展示研究成果
- 参加一个RL主题的竞赛，汇报比赛内容和比赛结果，总结参赛经历

附录-RL打怪升级之路

入门

在这个部分，一些RL的学习资源被推荐。它们具有不同的难度和风格，你需要选择适合自己的学习路线

课程/课件

- David Silver(UCL)
  以上是网课，这里是课件
  大神David Silver亲自授课，它仅包含RL最基础的理论部分，注重数学解析
- 李宏毅(NTU台大)
  以上是网课，这里是(部分)课件
  它覆盖知识较多和琐碎
- 周博磊(UCLA)
  这是Github中的课件资源，在README附中有网课资源
  它注重数学解析，提供高质量代码
- 百度(Baidu)
  这是百度提供的网课
  它手把手教学，有详细的算法和代码汉语讲解，相当基础，但覆盖知识少
- Spinning Up
  这是OpenAI提供的课程

它内容多而结构清晰，每个部分都有完整的概念、代码和文献清单

博客

- Hands-on-RL(SJTU)
  上交大强化学习课程材料，包含足够多的RL模型详解，提供完整代码，适合入门学习
- 深度强化学习实验室
  由深度强化学习实验室发起的项目，是各种课程、机构、竞赛等资源的汇总
  或许这一个项目用来学习就足够了

书籍

- Reinforcement Learning: An Introduction (By Richard S. Sutton and Andrew G. Barto)
  经典教材，包含RL基础部分，内容古老

进阶

在这个部分，一些有关RL的机构/网站被给出。如果你希望进行RL的科研/竞赛，就会更经常在社区和论文中学习，而且你需要的资料很可能并不在下文中

竞赛

- 及第
  这是一个汇总RL竞赛的平台
- Google Research Football with Manchester City F.C.
  这是一个谷歌和曼城合作的竞赛，它提供了一个开源环境以训练足球机器人
  Kaggle平台偶尔会出现RL比赛
- MineRL
  这是专门训练智能体玩Minecraft的竞赛，来自MineRL Labs
  它发起的BASALT Competition 2022正在进行中!
- AIcrowd
  AIcrowd，包括许多企业和高校发布的竞赛，一部分是RL竞赛，如前段时间刚结束的2nd Neural MMO challenge

社区

- RLChina
- 深度强化学习实验室
  两个国内强化学习社区，包含竞赛发布、企业招聘等信息

科研

RL相关的方向相当多，想必你已经有心仪的目标了

- Bandit
  Bandit算法社区，MAB问题是团队目前做的工作

其它

- stable-baselines3
- Tianshou
- garage
  这些是用PyTorch实现的RL算法包，可以辅助研究人员快速验证想法
- Gym
  这是OpenAI提供的RL开源环境Gym，被广泛用于RL研究
  你可以使用它训练自己的智能体，验证自己的RL算法
- DeepMind
- OpenAI
  两个著名研究机构，经常发布有趣的工作