作业 4：强化学习

清华大学软件学院
机器学习, 2023 年秋季学期

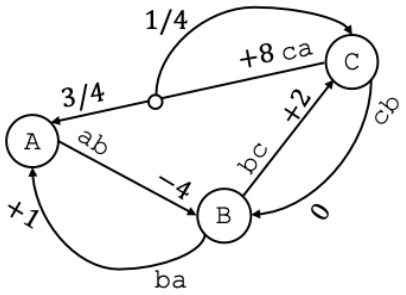
1 介绍

本次作业需要提交说明文档（PDF 形式）和 Python 的源代码。注意事项如下：

- 作业按点给分，因此请在说明文档中按点回答，方便助教批改。
- REINFORCE & AC 题目中使用的 Pytorch 主要用于计算梯度，使用 CPU 即可快速运行。
- 不要使用他人的作业，也不要向他人公开自己的作业，否则处罚很严厉，会扣至-100（倒扣本次作业的全部分值）。
- 统一文件的命名：{学号}_{姓名}_hw4.zip

2 Bellman Equation (20pt)

考虑下图所示马尔可夫决策过程（MDP）：衰减系数 $\gamma = 0.5$，大写字母 A、B、C 表示状态，小写字母组合 ca、ab、cb、bc、ba 表示可以采取的动作，正负整数表示采取行为可以获得的奖励，分叉分支上的分数表示转移概率。例如从状态 C 采取动作 ca 有 $\frac{3}{4}$ 概率可以到达状态 A，有 $\frac{1}{4}$ 概率依然在状态 C，两种情况均可以获得 +8 奖励。其他情况下，采取动作一定可以完成状态转移。



1. 写出衰减系数为 γ 的 MDP 中，策略 π 的状态值函数 $V^{\pi}(s)$ 的定义。

2. 写出状态值函数 $V^{\pi}(s)$ 所符合的贝尔曼（Bellman）期望方程。

3. 考虑一个均匀随机策略 π_0（以相同的概率选取所有动作），初始状态值函数 $V_0^{\pi_0}(A) = V_0^{\pi_0}(B) = V_0^{\pi_0}(C) = 0$，请利用 2 中的贝尔曼期望方程，写出上述 MDP 过程中，迭代式策略评估进行一步更新的状态值函数 $V_1^{\pi_0}$。

4. 基于 3 中计算得到的 $V_1^{\pi_0}$，利用贪心法得到确定性策略 π_1。

3 Monte Carlo & Temporal-Difference （15pt）

考虑未知环境的马尔可夫决策过程（MDP），其衰减系数 $\gamma = 1$，有两个状态 A,B，使用策略 π 采集到了下面两条轨迹：

$$B : b-2 \to A : a+3 \to B : b-3 \to \text{terminate} \tag{1}$$

$$A : a+3 \to A : a+2 \to B : b-4 \to A : a+4 \to B : b-3 \to \text{terminate} \tag{2}$$

其中 $A : a+3 \to A$ 表示当前状态 A，采取动作 a，获得奖励 +3，转移到下一个状态为 A。本题中算法学习率均为 0.1。

1. 使用计算首次访问（first-visit）的蒙托卡洛方法估计状态值函数 $V(A), V(B)$ 和状态动作值函数 $Q(A,a), Q(B,b)$。

2. 使用计算每次访问（every-visit）的蒙托卡洛方法估计状态值函数 $V(A), V(B)$ 和状态动作值函数 $Q(A,a), Q(B,b)$。

3. 使用时序差分方法 TD(0) 估计状态值函数 $V(A), V(B)$ 和状态动作值函数 $Q(A,a), Q(B,b)$。

4 Q-Learning & Sarsa （20pt）

Gym[1]是 OpenAI 开源的一套强化学习环境，用于开发和对比强化学习算法。在本题中，你将基于 Taxi-v3[2]环境，利用强化学习算法控制出租车实现乘客的接送。本题需要提交实验报告，代码见 `./code/sarsa_Q_learning`。**注意：请安装 0.19.0 版本的 gym**。

1. 补充 `./algorithms/QLearning` 函数，**填入 1 行代码**实现 Q-learning 算法；
2. 补充 `./algorithms/Sarsa` 函数，实现 Sarsa 算法；（可参考提供的 Q-learning 算法）
3. 完成不同算法迭代步长 `lr` 取值下的对比实验，对比结果用表格呈现，并用文本对结果进行分析。
4. 对比 Q-learning、Sarsa 实验效果（对比内容包括最终奖励值，完成任务所需动作数，算法迭代稳定性），并对结果进行分析。

[1] `https://gym.openai.com/`

[2] `https://gym.openai.com/envs/Taxi-v3/`

5 REINFORCE & AC （45pt）

在本题中，你将基于 CartPole-v0 和 CartPole-v1[3]环境，利用强化学习算法控制平衡木。本题需要提交实验报告，代码见 `./code/policy_gradient`。

1. 补充 `REINFORCE` 类中的 `learn` 函数，实现 REINFORCE 算法。
2. 补充 `TDActorCritic` 类中的 `learn` 函数，实现 TD Actor-Critic 算法。value 的损失函数已经预先实现了，只需要实现 policy 的损失函数即可。代码中 td_target $= R_{t+1} + \gamma v_\pi(S_{t+1})$。
3. 请绘制**两个模型**分别在**两个环境**上的训练曲线，包括训练过程的损失函数的变化和最终奖励值，并分析训练稳定性及收敛效率。由于强化学习的不稳定性，你的结果需要基于至少 3 个种子。

提示：

1. 动手之前，请仔细阅读代码中的注释，确保你已了解问题定义和代码框架。
2. 你可以解除位于 198 行的注释以获取可视化结果，env.render() 会渲染环境，让你看到平衡木的控制结果。
3. 本题已经提供两种 Policy Gradient 算法的代码框架，希望你完成损失函数部分，可以参考论文[4]与课件，REINFORCE 位于第 10 讲课件第 16 页，TD Actor-Critic 位于第 10 讲课件第 24 页，TD Actor-Critic 和 QAC 的核心思想是一致的，区别在于 critic 网络输出的不是 q 值，而是 value。如果你理解了 QAC，那么你应该可以很轻松地完成本次作业。
4. TD Actor-Critic 中，make_batch() 函数已经将所有需要用到的变量转换为 torch.Tensor，你可以直接调用 self.ac.v(self.states) 获取不同状态的价值 $v_\pi(S)$。
5. REINFORCE 算法在 cartpole-v1 上可能无法获取 500 的最终奖励，训练中出现抖动是正常现象。你只需要确保 REINFORCE 在 cartpole-v0 上获取 200 的最终奖励即可拿到分数。AC 算法应该可以在两个环境上都获取最高奖励，如果 3 个种子都没有获得最高奖励，也许你应该检查一下你的实现。
6. Pytorch 框架在本题中的用法等价于 numpy，比如可以通过 `torch.mean` 计算均值，通过 `torch.std` 计算方差。在 debug 过程中，如果你无法确定一个 Tensor 的形状，你可以使用 Tensor.shape 获取之。如果你之前没有安装过 Pytorch，推荐通过 conda 安装 cpu 版本，具体命令请参考[5]。
7. pytorch 在计算时会保存计算图，以供 autograd 模块自动求导。因此请注意在计算时不要使用 torch.tensor() 创建中间变量，因为这样创建出的变量是值拷贝，不在计算图上，不会计算梯度。可以使用 torch.cat() / torch.stack() / torch.gather() 创建中间变量。

[3] `https://www.gymlibrary.dev/environments/classic_control/cart_pole/`

[4] `https://homes.cs.washington.edu/~todorov/courses/amath579/reading/PolicyGradient.pdf`

[5] `https://pytorch.org/get-started/locally/`