Artificial Intelligence
人工智能实验

深度强化学习

中山大学计算机学院
2024年春季

目录

1.1 深度强化学习介绍

什么是深度强化学习？

- 传统的RL算法有个很大的问题在于它是一种表格方法，就是根据过去出现过的状态，统计和迭代Q值。这些基于表格的方法， 一方面适用的状态和动作空间非常小，对于图像和高维度离散状态、连续域状态无法直接适用；另一方面对于一个状态从未出现过，这些算法是无法处理的，也就是说基于表格的算法没有对未知状态的泛化能力。

- 深度模型（Deep）的引入使得强化学习算法（RL）解决更复杂的问题
  - Deep = can process complex sensory input，能用于处理复杂的感知输入
  - RL = can choose complex actions，能用于选择复杂的动作

- 深度神经网络用于状态值函数、动作值函数、策略的表征

1.2 DQN算法

Q-learning算法回顾

- 是一种 value-based 的强化学习算法，Q即为Q(s,a)，在某一时刻的state状态下，采取动作action能够获得收益的期望。
- 主要思想是将state和action构建一张Q-table表存储Q值，然后根据Q值选取能够获得最大收益的动作。
- 基于off-policy时序差分法，且使用贝尔曼方程可以对马尔科夫过程求解最优策略。
- 伪码：

1. Initialize Q-values ($Q(s, a)$) arbitrarily for all state-action pairs.
2. For life or until learning is stopped...
3. Choose an action (a) in the current world state (s) based on current Q-value estimates ($Q(s, \cdot)$).
4. Take the action (a) and observe the the outcome state (s') and reward (r).
5. Update $Q(s, a) := Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$

1.2 DQN算法

DQN算法

- 将Q-learning算法和深度神经网络结合，并额外引入两个机制**经验回放**和**目标网络**
- **经验回放（Replay Buffer）：**将智能体探索环境得到的数据储存起来，然后随机采样小批次样本更新深度神经网络的参数
  - 引入原因：
    - 深度神经网络作为有监督学习模型，要求数据满足独立同分布
    - Q-Learning 算法得到的样本前后是有关系的。为了打破数据之间的关联性，Experience Replay 方法通过存储-采样的方法将这个关联性打破了。
  - 优点：
    - 数据利用率高，因为一个样本被多次使用。
    - 连续样本的相关性会使参数更新的方差（variance）比较大，该机制可减少这种相关性。注意这里用的是均匀随机采样

1.2 DQN算法

DQN算法

- 将Q-learning算法和深度神经网络结合，并额外引入两个机制**经验回放**和**目标网络**
- **目标网络（Target Network）：**额外引入一个目标网络（和Q网络具有同样的网络结构），此目标网络不更新梯度，每隔一段时间将Q网络的参数赋值给此目标网络。
  - 引入原因：
    - 深度神经网络作为有监督学习模型，要求监督数据标签是稳定的
    - Q-Learning算法使用下一时刻的Q值和奖励值作为监督信号，由于每次神经网络更新后，Q值会变化，导致Q-Learning算法的监督信号不稳定。
  - 优点：
    - 一定程度降低了当前Q值和目标Q值的相关性。
    - 在一段时间里目标网络的Q值是保持不变的，提高了算法稳定性。

1.2 DQN

DQN算法

- 不加经验回放和目标网络的DQN，通常被称为Naïve DQN，伪码如下：

算法 2.1 Naive DQN 算法[66]

Input: 样本数据 $\mathcal{B}$

Output: 策略 π

随机初始化 Q 值网络参数 ϕ，初始化学习率 α，最大回合数 N，一个回合的最大时间步 T

for 每个回合 $ep = 1, 2, .., N$ **do**

  重置环境并获取环境的初始状态

  for 每个时间步 $t = 1, ..., T$ **do**

    在时刻 t，状态为 s 时，智能体根据 ϵ 贪心策略和所有动作的值函数选择对应的动作 a

    根据状态转移函数，环境转移到下一时间步状态 s' 并返回对应的奖励值 r

    使用公式 (2-15)更新神经网络参数 ϕ

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s,a)\left(Q_\phi(s,a) - y\right),$$

$$y = r(s,a) + \gamma \max_{a'} Q_\phi(s',a'),$$

  end for

end for

1.2 DQN

DQN算法

- 加经验回放和目标网络的DQN，伪码如下：

算法 2.2 DQN 算法[66]

Input: 经验回放池 $\mathcal{B}$，批样本大小 B，目标网络更新间隔 d

Output: 策略 π

随机初始化 Q 值网络参数 ϕ，初始化学习率 α，最大回合数 N，一个回合的最大时间步 T，梯度更新次数 G
初始化目标值网络参数 $\phi' \leftarrow \phi$
for 每个回合 $ep = 1, 2, .., N$ **do**
  重置环境并获取环境的初始状态
  for 每个时间步 $t = 1, ..., T$ **do**
    在时刻 t，状态为 s 时，智能体根据 ϵ 贪心策略和动作的值函数选择对应的动作 a
    根据状态转移函数，环境转移到下一时间步状态 s' 并返回对应的奖励值 r
    将四元组 (s, a, r, s') 存入经验回放池 $\mathcal{B}$ 中
    for 每个梯度更新步 $g = 1, ..., G$ **do**
      从经验回放池 $\mathcal{B}$ 中采样一批大小为 B 数据样本
      使用公式 (2-17)更新神经网络参数 ϕ
    end for
    if t 除以 $d == 0$ **then**
      使用公式 (2-18)更新目标网络参数 ϕ'
    end if
  end for
end for

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s,a)\left(Q_\phi(s,a) - r(s,a) - \gamma \max_{a'} Q_{\phi'}(s',a')\right),$$

$$\phi' \leftarrow \phi, \ every\ d\ time\ steps.$$

2. 实验任务（无需提交）

- CartPole任务
  - 在CartPole环境中实现DQN算法。
  - 要求：
    - 在给定的代码框架下补充代码。
    - 最终的reward至少收敛至180.0。