姓名：胡华金　　学号：22351080
学院（系）专业：软件学院　软件工程

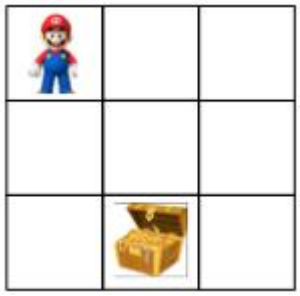
一．算法描述

算法历史演变：

1. Value Iteration 值迭代

设我们有一个 3 x 3 的棋盘：

有一个单元格是超级玛丽，每回合可以往上、下、左、右四个方向移动
有一个单元格是宝藏，超级玛丽找到宝藏则游戏结束，目标是让超级玛丽以最快的速度找到宝藏
假设游戏开始时，宝藏的位置一定是(1, 2)



这个一个标准的马尔科夫决策过程(MDP)：

状态空间 State：超级玛丽当前的坐标

决策空间 Action: 上、下、左、右四个动作

Action 对 State 的影响和回报 P(State', Reward | State, Action)：本文认为该关系是已知的

超级玛丽每移动一步，reward = -1

超级玛丽得到宝箱，reward = 0 并且游戏结束

2. Policy Iteration 策略迭代

利用策略迭代 (Policy Iteration) 求解马尔科夫决策过程

上一篇我们介绍了如何使用价值迭代 (Value Iteration) 来求解 MDP，这篇介绍另外一种方法：策略迭代 (Policy Iteration)。

关于策略迭代，需要知道下面几点：

什么叫策略？策略就是根据当前状态决定该采取什么 Action，
- P(Action | State)

以本文超级玛丽寻找宝箱为例，超级玛丽需要不断朝着宝箱的方向前进

当前状态在宝箱左侧，策略应该是朝右走

当前状态在宝箱上方，策略应该是超下走

如何衡量策略的好坏 —— 策略评估 (Policy Evaluation)

还记得上一节定义的价值函数 (Value Function) 吗？给定一个策略，我们可以计算出每个状态的期望价值 V(s)

如何找到更好的策略 —— 策略迭代 (Policy Iteration)

初始化：随机选择一个策略作为初始值，比如说不管什么状态，一律朝下走，即 P(Action = 朝下走 | State) = 1，P(Action = 其他 Action | State) = 0

第一步 策略评估 (Policy Evaluation)：根据当前的策略计算 V(s)

第二步 策略提升 (Policy Improvement)：计算当前状态的最好 Action，更新策略，

不停的重复策略评估和策略提升，直到策略不再变化为止

3. Monte Carlo Learing 蒙特卡洛学习

动态规划方法是建立在模型已知的情况下，但是往往大多数情况下模型是未知的，实际应用中我们不可能完全了解一个环境的所有知识，比如说得出它的状态转移矩阵。这个时候蒙特卡洛算法就派上用场了，它只需要从经验（experience）中去学习，这个经验包括样本序列的状态（state）、动作（action）和奖励（reward）。得到若干样本的经验后，通过平均所有样本的回报（return）来解决强化学习的任务。

类似于 DP 方法，MC 求解也可以看作是一种广义的策略迭代过程，即先计算当前策略所对应的值函数，再利用值函数来改进当前策略，不断循环这两个步骤，从而得到最优值函数和最优策略。

4. Q-learning

Q-Learning 是一种异策略(off policy)的时序差分方法，即动作策略为 ε -greedy 策略，目标策略为贪婪策略。在更新值函数时并不完全遵循交互序列，而是选择来自其他策略的交互序列的子部分替换了原来的交互序列。从思想来说，它结合了子部分的最优价值，更像是结合了价值迭代的更新算法，希望每一次都使用前面迭代积累的最优结果进行更新。

5. DQN（Deep Q-learning Network）

DQN 是一种深度增强学习算法，它采用神经网络来学习 Q 值函数。Q 值函数是一个将状态和行动映射到 Q 值的函数，表示通过执行该行动在特定状态下获得的预期回报。这里的 Q 值函数是使用深度神经网络进行建模的，因此被称为 Deep Q Networks，简称 DQN。

6. Sarsa

Sarsa 全称是 state-action-reward-state'-action'。 也是采用 Q-table 的方式存储动作值函数；而且决策部分和 Q-Learning 是一样的，也是采用 ε -greedy 策略。不同的地方在于 Sarsa 的更新方式是不一样的。

1.Sarsa 是 on-policy 的更新方式，它的行动策略和评估策略都是 ε -greedy 策略。

2.Sarsa 是先做出动作后更新。

Q-Learning 算法，先假设下一步选取最大奖赏的动作，更新值函数。然后再通过 ε -greedy 策略选择动作。

Sarsa 算法，先通过 ε -greedy 策略执行动作，然后根据所执行的动作，更新值函数。

7. Policy Gradient

PG 是一个蒙地卡罗+神经网络的算法。

8. Actor-Critic

为了解决 High Variance 和 High bias 之间的矛盾，可以把它们结合在一起，利用 value based 和 policy based 两类方法各自的优势，还顺带把它们的短板都补上了。于是就有了集大成的 Actor-Critic 类方法。 具体来说，就是构造一个全能型的 agent，既能直接输出策略，又能通过 value function 来实时评价当前策略的好坏。所以我们需要两个网络，一个负责生成策略的 Actor 和一个负责评价策略的 Critic。这就有点类似一个演员在表演，

而同时一个评论家在随时纠正他的表现，而且两者都还在不断更新，这种互补式的训练方式会比单独的策略网络或者值函数网络更有效。

9. DDPG（Deep Deterministic Policy Gradient）

Deterministic Policy 是相对于 Stochastic Policy 而言的。其中 Stochastic Policy 的表达式为 $\pi_\theta(a|s) = P[a|s;\theta]$，在实际应用中，大家往往采用高斯分布来作为策略的分布。其中高斯分布的均值和方差都由神经网络来近似。

而 Deterministic Policy 的表达式为 $a = \mu_\theta(s)$ ，给定一个状态 s 就会对应一个唯一动作 a。

Deterministic Policy 有什么好处？

从实际的角度，对于 stochastic 情况，策略梯度需要对状态和动作同时积分，而 deterministic 情况只需要对状态积分。这个不同造就了 deterministic 相比于 stochastic 策略在高维动作空间的时候更容易训练。

10. A3C（Asynchronous Advantage Actor-Critic）

A3C 算法和 DDPG 类似，通过 DNN 拟合 policy function 和 value function 的估计。

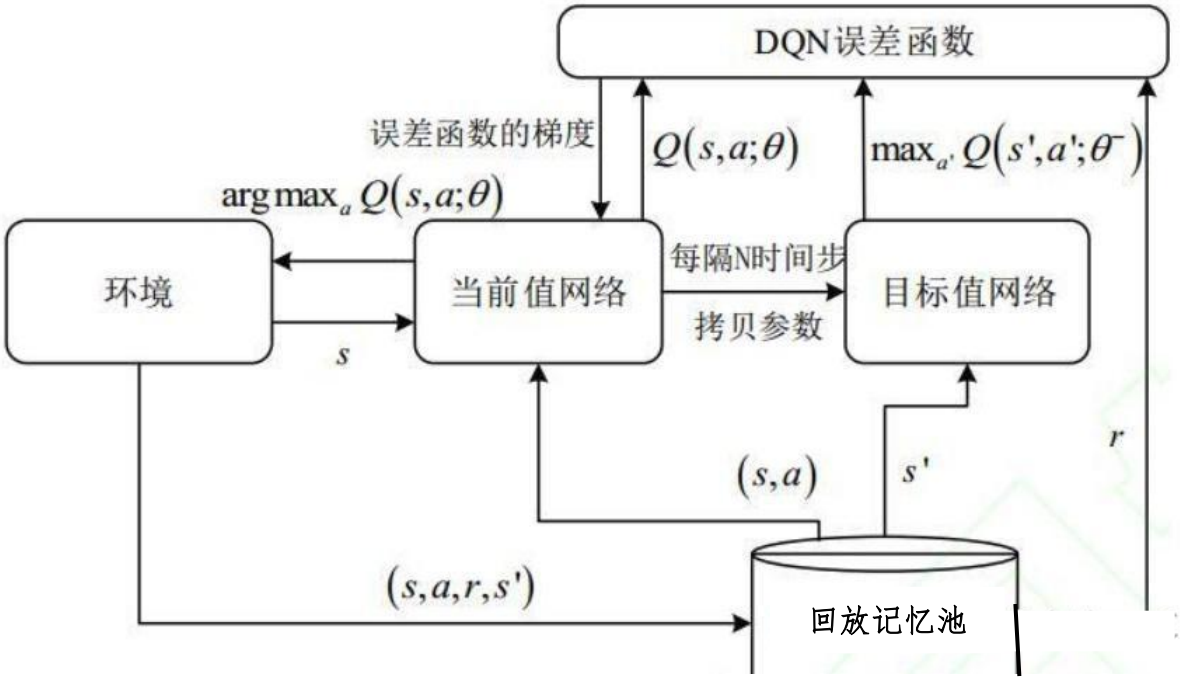
但是不同点在于:

. A3C 中有多个 agent 对网络进行 asynchronous update，这样带来了样本间的相关性较低的好处，因此 A3C 中也没有采用 Experience Replay 的机制；这样 A3C 便支持 online 的训练模式了

. A3C 有两个输出，其中一个 softmax output 作为 policy $\pi(a_t|s_t;\theta)$，而另一个 linear output 为 value function $V(s_t;\theta_v)$ ，其余 layers 都共享。

. A3C 中的 Policy network 的评估指标采用的是上面比较了多种评估指标的论文中提到的 Advantage Function(即 A 值) 而不是 DDPG 中单纯的 Q 值。

算法流程图：



二. 算法性能分析

Q-Learning

在值函数方法中的Q-Learning算法的大致流程：

1.初始化网络 $Q_\phi(s,a)$

2.使用贪心策略采集一些轨迹$\{s_i,a_i,r_i,s_i'\}$

3.计算 $y(s_i,a_i)\approx r(s_i,a_i)+\gamma\max_{a'}Q_\phi(s_i',a_i')$

4.更新参数 $\phi,\ \phi <- \phi - \alpha \sum_i \frac{dQ_\phi(s_i,a_i)}{d\phi}(Q_\phi(s_i,a_i) - y(s_i,a_i))$ 返回直到收敛

Deep Q-Network(DQN)

两个非常重要的思想：经验回放和目标网络 （1） Experience Replay，其将系统探索环境得到的数据储存起来，然后随机采样样本更新深度神经网络的参数。

Experience Replay 的原因：

1、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

2、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。

在这个问题中，之所以加入 experience replay 是因为样本是从游戏中的连续帧获得的，这与简单 RL 比如 maze）相比，样本的关联性大了很多，如果没有 experience replay，算法在连续一段时间内基本朝着同一个方向做梯度下降，那么同样的步长下这样直接计算 gradient 就有可能不收敛。因此 experience replay 是从一个 memory pool 中随机选取了一些 experience，然后再求梯度，从而避免了这个问题

Experience Replay 优点：

1、数据利用率高，因为一个样本被多次使用。

2、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

Deep Q-Network(DQN)与 Q-Learning 相比

DQN 主要改进在以下三个方面：

（1）DQN 利用深度卷积网络(Convolutional Neural Networks,CNN)来逼近值函数；

（2）DQN 利用经验回放训练强化学习的学习过程；

（3）DQN 独立设置了目标网络来单独处理时序差分中的偏差

成立，模型的效果就会大打折扣。而上面提到的相关性恰好打破了独立同分布的假设，那么学习得到的值函数模型可能存在很大的波动。

三．算法进一步研究展望

深度学习的神经网络是建立在 iid(样本独立同分布)的基础上，本质是采用概率统计的方式去拟合输入和输出的关系。因此，在遇到 ood(out of distribution 分布之外）的数据的时候，神经网络将无法处理而输出错误的结果。

由于这个问题的存在，导致深度学习很难处理 corner case！深度强化学习是在深度学习基础上结合强化学习发展出来的技术，同样有这样的问题。因此，在现实落地场景中，有很多场景是要求 99.99%甚至更高的准确度的，比如机械臂的抓取，比如自动驾驶。那么在这种场景下，落地要求对 corner case 能够很好的处理，就会变得很难。

要解决这个问题，要么深度学习理论本身发生变革，能够很好的支持 ood，要么只能在数据端进行处理，让智能体的训练尽可能的覆盖各种各样的场景，让 corner case 尽可能的少。

要变革深度学习理论是一件非常困难的事情，因此，目前的做法就是通过更多样的数据来处理。

数据的多样性就成了深度强化学习需要解决的难题。就算在虚拟场景的落地比如游戏 AI，一样会遇到这个问题。

深度强化学习的未来

在工业界，期待深度强化学习在全领域（虚拟与现实场景）进行大规模的落地。

深度强化学习不同于一般深度学习，核心是决策！为什么游戏 AI 需要深度强化学习？因为基于行为树的 AI 远远弱于基于深度强化学习的 AI。同样的，在很多现实场景，基于人类经验的操作是很差的，通过深度强化学习可以实现大幅度的提升。在这种场景中，深度强化学习就是刚需。

由于深度强化学习目前还存在的问题，要落地是需要进一步的技术突破的，包括但不限于更大的规模（数据和计算资源）。

比如深度强化学习要在机器人上进行落地，就需要非常拟真的虚拟环境进行训练，同时面向传统方法通过规划很难解决的问题，比如收拾家务。这和自动驾驶的问题很像，要 L4 也是

需要深度强化学习的（通过规则已很难处理过于复杂的场景），但同样需要很好的虚拟环境支持，去覆盖 99.99%的场景。