浙江大學
ZHEJIANG UNIVERSITY

人工智能算法与系统-2024 秋

机器人自动走迷宫

姓名 刘京宗

学号 22451040

学院 软件学院

2024 年 10 月 18 日

1 实验介绍

本实验旨在通过实现两种不同的算法——基础搜索算法和 Deep Q-Learning 算法，来引导机器人在一个迷宫环境中自动找到出口。该实验不仅考察了传统搜索算法在路径规划中的表现，也通过强化学习中的深度 Q-Learning 来展示在复杂环境中机器学习算法的优越性。

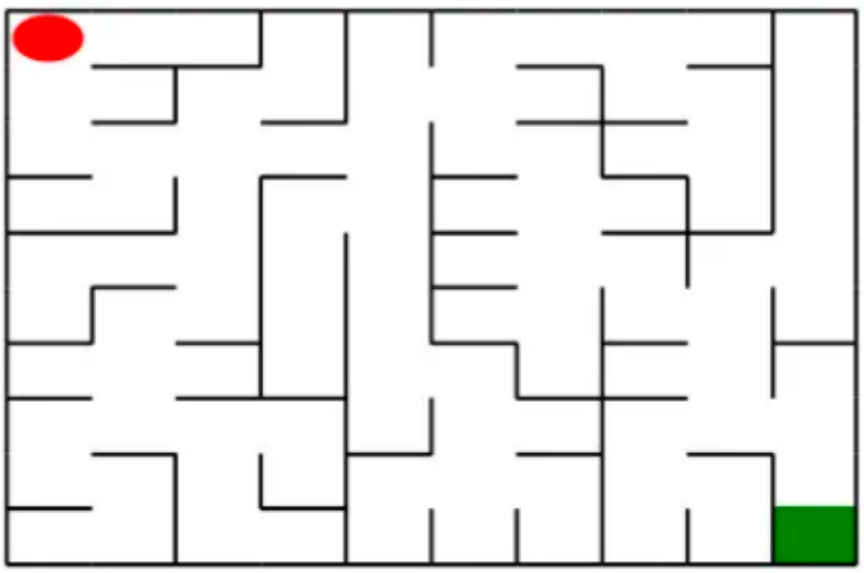

图 1: 迷宫初始状态，标注起点和终点

起点：迷宫左上角的红色椭圆终点：迷宫右下角的绿色方块游戏规则：机器人每次只能选择一个方向（上、下、左、右），并在迷宫中移动，目标是在最少步数内从起点到达终点。

2 基础搜索算法

基础搜索算法包括广度优先搜索（Breadth-First Search, BFS）和深度优先搜索（Depth-First Search, DFS），它们是经典的图搜索算法，常用于寻找路径、遍历节点。特别是在迷宫问题中，这些算法可以用于找到从起点到终点的可行路径或最短路径。

2.1 广度优先搜索（BFS）

广度优先搜索是一种逐层扩展的搜索算法，它的基本思想是从起点开始，按照距离逐层扩展迷宫中的节点，直到找到目标点（终点）。它是一种无权图中的最短路径搜索算法，保证能够找到最短路径。

2.1.1 BFS 算法步骤

BFS 算法的具体步骤如下：

1. 初始化一个队列（Queue），将起点加入队列，并标记为已访问。
2. 当队列非空时，执行以下操作：
   - 从队列中取出队首节点，记为当前节点。
   - 检查当前节点是否为目标节点（终点），如果是，则返回该节点的路径。
   - 否则，将当前节点的所有未被访问过的邻居节点加入队列，并标记为已访问。
3. 重复步骤 2，直到找到目标节点或队列为空。

2.1.2 BFS 算法的特点与适用场景

- 保证最短路径：BFS 会按距离逐层扩展，因此它保证找到从起点到终点的最短路径。特别是在迷宫问题中，BFS 可以找到一个从起点到终点的最短步数路径。
- 空间复杂度较高：由于 BFS 需要存储所有已访问节点和未访问节点，空间复杂度相对较高，特别是在大规模迷宫中，可能需要较大的内存来存储扩展的节点队列。
- 适用场景：BFS 适用于节点较少且路径较短的迷宫，因为它在扩展过程中考虑每一层的所有可能路径，适合用于迷宫中寻找最短路径的情形。

2.2 深度优先搜索（DFS）

深度优先搜索是一种沿着迷宫中的某一条路径一直探索，直到到达终点或无法继续为止的算法。如果遇到死胡同，它会回溯到上一个选择点并尝试另一条路径。DFS 不保证找到最短路径，但在某些情况下可能更快找到一条可行路径。

2.2.1 DFS 算法步骤

DFS 算法的具体步骤如下：

1. 初始化一个栈（Stack），将起点压入栈中，并标记为已访问。
2. 当栈非空时，执行以下操作：
   - 从栈顶取出当前节点。
   - 检查当前节点是否为目标节点（终点），如果是，则返回该节点的路径。
   - 否则，将当前节点的所有未被访问过的邻居节点压入栈，并标记为已访问。
3. 重复步骤 2，直到找到目标节点或栈为空。

2.2.2 DFS 算法的特点与适用场景

- 不保证最短路径：DFS 沿着一条路径一直走到底，如果找到终点，它返回的可能是较长的路径，不保证是最短路径。
- 空间复杂度较低：DFS 只需要存储当前路径的节点，因此在空间复杂度上优于 BFS，特别是在树状结构或者路径较深的迷宫中表现更优。
- 适用场景：DFS 更适合于探索较深的迷宫，或者当路径复杂度较高时，可以更快找到一条可行路径。然而在一些较复杂的迷宫中，DFS 可能会因为过度回溯而效率较低。

3 强化学习算法介绍

强化学习作为机器学习算法的一种，其模式是让智能体在“训练”中学到“经验”，以实现给定的任务。但不同于监督学习与非监督学习，在强化学习的框架中，更侧重通过智能体与环境的交互来学习。通常在监督学习和非监督学习任务中，智能体往往需要通过给定的训练集，并辅之以既定的训练目标（如最小化损失函数）来实现这一目标。然而，在强化学习中，智能体是通过其与环境的交互得到的奖励来学习。这个环境可以是虚拟的（如虚拟的迷宫），也可以是真实的（如自动驾驶汽车在真实道路上收集数据）。

在强化学习中有五个核心组成部分：环境（Environment）、智能体（Agent）、状态（State）、动作（Action）和奖励（Reward）。

在某一时间点 t，智能体的操作流程如下：

- 智能体感知其所处的状态 s_t
- 智能体根据某些准则选择动作 a_t
- 环境根据智能体的动作反馈奖励 r_{t+1}

通过合理的学习算法，智能体可以学到一个策略 $\pi(s_t) = a_t$，该策略决定在状态 s_t 下选择的动作 a_t。

4 QLearning 算法

Q-Learning 是一种值迭代（Value Iteration）算法，与策略迭代（Policy Iteration）算法不同，它计算每个状态或状态-动作对的值（Value）或效用（Utility），并在执行动作时最大化这个值。Q-Learning 通过估计每个状态的长期回报来找到最优策略，不仅考虑当前动作带来的奖励，还考虑未来的回报。

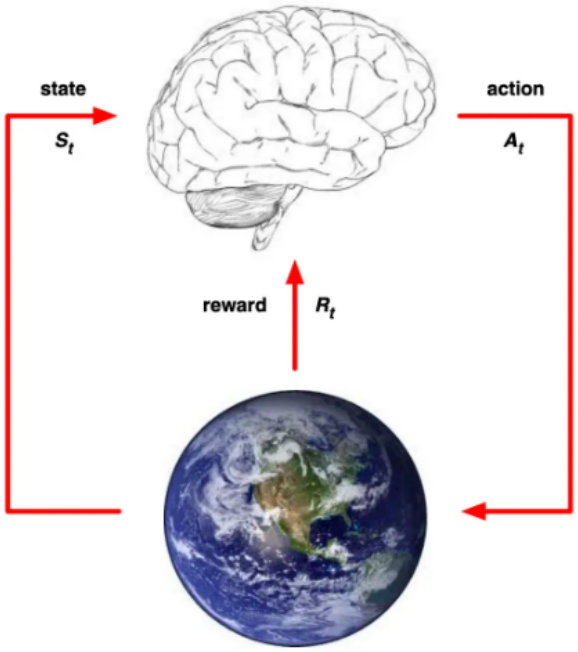

图 2: 强化学习的结构图

4.1 Q 值的计算与迭代

Q-Learning 算法使用 Q 表（Q-Table）存储每个状态-动作对的 Q 值。Q 表的行代表状态，列代表动作。

Q-Table	a_1	a_2
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$

图 3: Q 表示例

Q 值的更新公式为：

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

其中：

- s_t：时间 t 时的状态

- a_t：在时间 t 采取的动作
- r_{t+1}：时间 $t+1$ 时获得的即时奖励
- α：学习率
- γ：折扣因子，表示未来奖励的重要程度

4.2 Q-Learning 算法的实现步骤

Q-Learning 算法的主要步骤如下：

1. 初始化 Q 表
2. 在每个时间步，智能体根据当前状态选择动作 a_t
3. 执行动作，获得即时奖励 r_{t+1} 并更新 Q 表
4. 继续迭代直到收敛

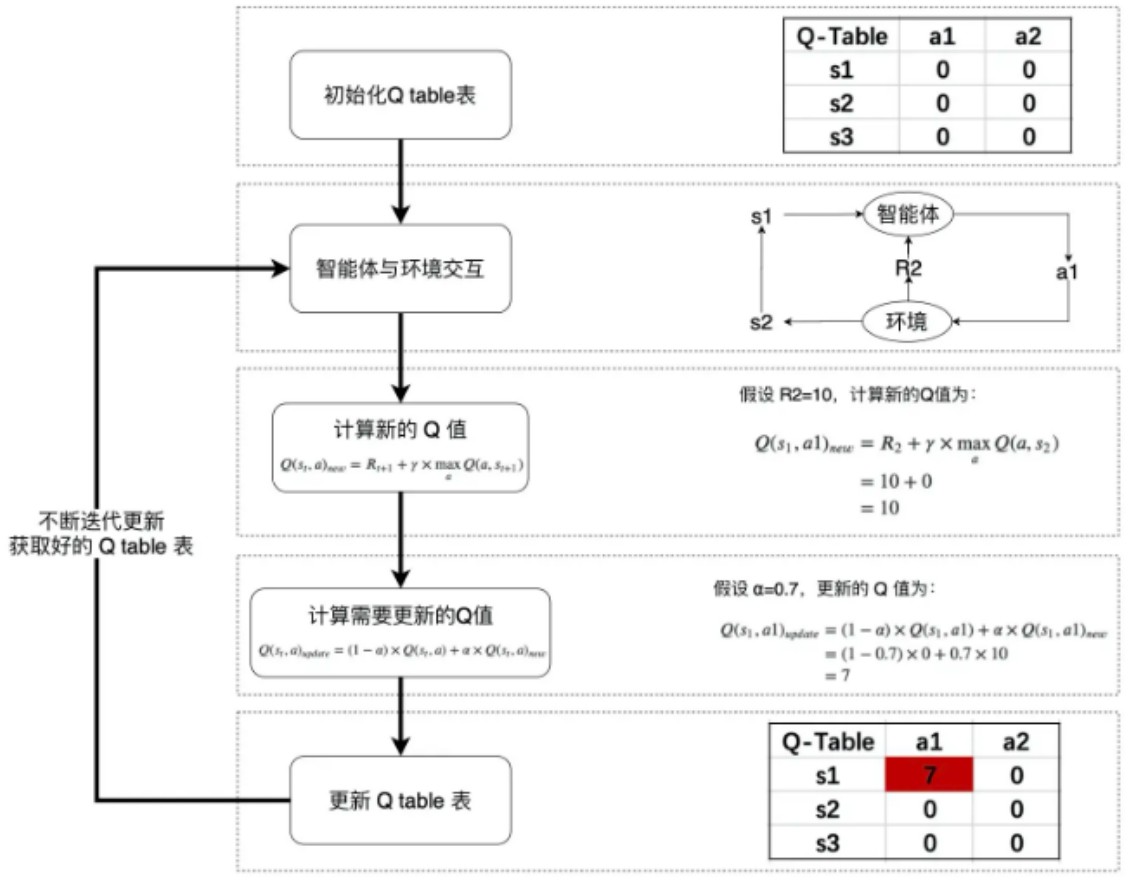

图 4: Q-Learning 算法的流程示意图

5 实验结果与分析

图 5: 实验结果

通过实验，我们可以得出以下结论：

1. 基础搜索算法适合小规模、规则化的迷宫，能够保证找到解，但在复杂迷宫中可能效率不高。

2.Deep Q-Learning 算法在面对复杂的迷宫时表现出色，能够通过反复学习找到最优解，尽管初期学习阶段需要更多的时间。

3. 在实际应用中，对于规模较大的迷宫或变化频繁的环境，Deep Q-Learning 等智能算法更具优势。