

文章编号:1007-9432(2021)01-0091-07

# 改进的 $Q$ -Learning 算法及其在路径规划中的应用


毛国君, 顾世民

(福建工程学院 机器学习与智能科学研究所, 福州 350118)

**摘 要:**在传统的  $Q$ -学习算法上, 提出了一种改进算法  $\epsilon$ - $Q$ -Learning, 并应用到路径规划中。引入了动态搜索因子, 其根据环境的反馈来动态调整贪婪因子  $\epsilon$ , 如果一次从起点到终点的探索失败, 则通过增大  $\epsilon$  来使下一次探索的随机性增大, 以免陷入局部优化困境; 反之, 则通过减少  $\epsilon$  来增加目的性。本实验利用算法损失函数、运行效率、步数、总回报来评估算法表现。实验表明,  $\epsilon$ - $Q$ -Learning 算法相比于已有的  $Q$ -Learning 算法, 不仅可以找到更优的路径, 而且可以有效地减少迭代搜索的代价。

**关键词:**路径规划; 人工智能; 强化学习;  $Q$ -Learning

**中图分类号:**TP391      **文献标识码:**A

**DOI:**10.16355/j.cnki.issn1007-9432tyut.2021.01.012      **开放科学(资源服务)标识码(OSID):** 

## An Improved $Q$ -Learning Algorithm and Its Application in Path Planning

MAO Guojun, GU Shimin

(Institute of Machine Learning and Intelligent Science, Fujian University of Technology, Fuzhou 350118, China)

**Abstract:** Traditional  $Q$ -Learning algorithm has the problems of too many random searches and slow convergence speed. Therefore, in this paper an improved  $\epsilon$ - $Q$ -Learning algorithm based on traditional  $Q$ -Learning algorithm was proposed and applied to path planning. The key of this method is to introduce the dynamic search factor technology, which adjusts the greedy factor dynamically according to the feedback of the environment. If one exploration from the beginning to the end fails, the randomness of the next exploration will be increased by increasing greedy factor, in order to avoid falling into the local optimization dilemma. Conversely, purpose will be increased by reducing greedy factor. The performance of the algorithm is evaluated by loss function, running efficiency, number of steps, and total return. Experiments show that compared with the existing  $Q$ -Learning algorithm,  $\epsilon$ - $Q$ -Learning can not only find a better optimal path, but also significantly reduce the cost of iterative searching.

**Keywords:** path planning; artificial intelligence; reinforcement learning;  $Q$ -Learning

人类知识的获取是通过感知环境 (Environment) 并与之交互来完成的, 因此所谓的智能学习就是智能体 (Agent) 不断适应环境来完善自己的过程。强化学习 (reinforcement learning, RL) 是机器学习的方法论之一, 用于描述和解决智能体在与环

境的交互过程中如何学习和优化策略的问题<sup>[1]</sup>。事实上, 许多应用中的环境是动态变化和不确定性的, 如机器人的路径规划问题, 智能体必须在不确定性环境中主动地对周围环境进行探索<sup>[2-3]</sup>。在探索过程中不断地认知环境, 形成适应环境的正确行为。

收稿日期: 2020-08-26

基金项目: 国家自然科学基金资助项目 (61773415)

通信作者: 毛国君 (1966—), 博士, 教授, 主要从事数据挖掘、机器学习及大数据研究, (E-mail) maximmao@hotmail.com

引文格式: 毛国君, 顾世民. 改进的  $Q$ -Learning 算法及其在路径规划中的应用[J]. 太原理工大学学报, 2021, 52(1): 91-97.

强化学习不同于目前广泛研究的监督学习(supervised learning)和无监督学习(unsupervised learning),它的学习不是被动地从已有数据中进行归纳或提取,而是一个主动适应环境并进行自我完善的过程。强化学习是从计算机科学、数学、神经学等多个相关学科发展而来的,已经成为机器学习的主要分支之一<sup>[4-5]</sup>。由于强化学习强调在环境变化情况下智能体的主动学习,因此近年受到广泛关注,并在机器人控制以及智能计算等领域得到应用。

基本的强化学习思想是通过智能体对当前环境状态的感知,并对可能采取的动作(Action)获得的回报(Reward)进行评价来完成的。图 1 给出了强化学习的基本模型<sup>[6]</sup>。



图 1 强化学习的基本模型

Fig. 1 Reinforcement learning model

强化学习是一个“探索-利用”的迭代过程<sup>[6-7]</sup>。首先,智能体通过感知环境的当前状态,并采取某一个动作来探索环境,探索的结果一般以某种形式的奖励或者回报来表示。然后,对已经获得的回报进行评价,寻找在当前状态下的一个最优的动作加以利用。当然,“探索-利用”是一个不断反复循环的过程,直到获得满意的最终策略。

不同的强化学习算法在“探索”“利用”方法及其融合机制上会有所差异。就强化学习的经典算法 Q-Learning 而言,在探索阶段所用的方法是  $\epsilon$ -贪心( $\epsilon$ -greedy)方法,即优先利用最大 Q 值对应的动作来向前推进探索。

**传统的 Q-Learning 存在探索效率低下的致命问题<sup>[8]</sup>。**实际上,探索与利用是强化学习的两个不可分割的部分,同时也是一对矛盾共同体。当智能体过多地探索环境时必然会导致效率下降,但当智能体在探索不充分情况下盲目地相信某种决策而加以利用时也会增加后续的探索代价。因此,制定一个合适的探索计划来平衡探索和利用的代价,以便在正确决策的同时尽可能地提高收敛速度,是强化学习的一个重要任务。

**本文针对传统的 Q-Learning 算法收敛速度慢的问题,提出一个改进算法  $\epsilon$ -Q-Learning。基本思想是通过动态地改变搜索因子参数来快速适应环境的变化,既可以提高探索效率,也可以避免陷入无效的**

**局部搜索,因而提高全局优化的可能性。**

## 1 相关工作

路径规划(Path planning)是机器人研究领域的一个关键性的工作,同时也是许多应用(如电子游戏、无人驾驶等)的基础。1986 年,Khatib 第一次提出用人工势场法(Artificial potential field method)解决了机器人躲避障碍物的问题,此后这个方法也成为许多路径规划研究与应用的基础<sup>[9]</sup>。然而,随着人工智能技术的研究进展,利用智能学习方法来解决路径规划问题成为新的发展趋势,其中强化学习扮演了十分重要的角色。例如:利用 BP 神经网络和 Q-Learning 算法,在未知环境下的自主避障问题被研究<sup>[10]</sup>;将模糊逻辑引入到强化学习中<sup>[11]</sup>;Agent 的自主强化学习等<sup>[12-14]</sup>。我国学者也在机器人、无人机等强化学习模型方面开展了卓有成效的工作<sup>[15-16]</sup>。

此外,强化学习的模型或者算法基本都是基于马尔可夫(Markov)属性构造的<sup>[4-5]</sup>。马尔可夫属性是指一个系统下的状态只与当前状态有关,而与更早之前的状态无关,即公式(1):

$$P(s_{t+1} | s_t, s_{t-1} \dots s_1) = P(s_{t+1} | s_t) \quad (1)$$

基本的马尔可夫决策(MDP)<sup>[4]</sup>模型由一个四元组  $\langle S, A, P, R \rangle$  来刻画:

其中,S 表示环境中的所有状态集合;A 表示是作用于环境的所有可能动作集合;P 表示状态之间的转移概率;R 表示采取某一动作到达下一状态后的回报。

基于 MDP 模型,目前的强化学习算法主要是围绕着回报的评价机制、状态与动作的影响估算等方面进行研究和实践,因而出现了不同的解决模型,也在算法的可用性等方面进行有效探索<sup>[17-18]</sup>。

目前的典型强化学习算法大致有 3 种,分别为 Q-Learning 算法、Sarsa 算法和 TD 算法<sup>[4,16-17]</sup>。分析这些典型算法的理论基础,基本上是由 2 个实体和 4 个评价机制组成的。当然在不同的强化学习算法中可能依赖的评价机制或者函数有所侧重。

实体主要有环境和智能体。

1) 环境是学习的对象。一般而言,在一个确定时刻,一个环境一定有一个确定的状态(State),但是当智能体在该环境中活动后,其状态就会发生改变。因此智能体必须对其活动结果(下一个状态)有一个估算,并借此形成下一步的决策。

2) 智能体则是学习者。一般而言,智能体是通过采取动作(Action)来适应环境的。就是说,智能

体需要通过不断地尝试某状态下可能动作的效果,来认知环境并采取恰当的动作来继续探索。

评价机制包括 4 个基本方面。

### 1.1 策略 $\pi$

在强化学习过程中,智能体在某个状态采取什么样的动作到下一个状态是由策略控制的。简单地说,策略就是从状态到动作的映射。特别地,当环境存在障碍或者陷阱时,策略就必须保证下个动作不能碰到障碍或者掉入陷阱。策略的好坏决定了智能体行动的好坏,进而决定了整个算法的学习质量。

### 1.2 回报 $R(s)$

回报  $R(s)$  是智能体处在状态  $s$  下可能形成正确决策的可能性:可能性大则回报值就大,反之亦然。强化学习的任务就是不断地探索来改变状态,达到寻优目的。因此,一个状态  $s$  的回报是在不断地探索中加以完善的。

给定一个探索的时间序列  $\langle 1, 2, \dots, t, t+1, \dots \rangle$ , 设当前时间为  $t$ , 根据马尔可夫属性,则  $R(s)$  的理论值就是:

$$R_t(s) \leftarrow f(R_t(s), R_{t+1}(s), \dots). \quad (2)$$

其中,  $R_t(s)$  为状态  $s$  在  $t$  时刻的回报,  $f$  则代表对向前探索的回报值的综合评价函数。

基于回报估计是强化学习的理论基础,也是构建强化学习算法的基本依据。当然,不同的算法在实现路径上会有不同,在回报值的计算及其评价函数的设计上会有所区别<sup>[19]</sup>。

从公式(2)可以看出,一个状态  $s$  的理论回报值的计算是一个反复探索直到稳定的过程。特别地,当环境的状态空间很大或者情况复杂的情况下,这种探索可能耗时很长、甚至是无限的。因此,实际的算法经常使用的是有限  $T$  步的探索策略。此外,在回报值的综合评价函数方面,最常用的方法是“折扣累计回报”,即给定探索的步数  $T > 0$  和折扣因子  $\gamma \in [0, 1]$ , 基于有限折扣累计回报策略,  $t$  时刻的状态回报值计算如下:

$$R(s) \leftarrow R_t(s) + \gamma \times R_{t+1}(s) + \dots + \gamma^T \times R_{t+T}(s). \quad (3)$$

### 1.3 状态值函数 $V(s)$

如上所述,环境的变化在强化学习中表现为状态的更新。值函数将理论上的回报值转化成可以计算的  $V$  值、并通过反复迭代来实现强化学习的目标。基于  $V$  值学习的模型和算法已经成为强化学习的一个重要方法。

给定一个时间  $t$  和当前状态  $s_t$ , 值函数方法是一个“前探+回推”的过程。基于有限折扣累计回报

策略,下面的公式(4)给出了对应值函数的计算方法:

$$V_{\pi}(s_t) \leftarrow R(s_t) + \sum_{k=1}^{T-1} \gamma^k \times P(s_{t+k}, s_{t+k+1}) \times V_{\pi}(s_{t+k}). \quad (4)$$

其中:  $\pi$  是学习的策略;  $R(s_t)$  是  $t$  时刻的回报(也被称为  $t$  时刻的即时回报);  $P(s_1, s_2)$  为状态  $s_1$  可能转移到状态  $s_2$  的概率;  $\gamma \in [0, 1]$  是折扣因子,而且整个后面一项是前探  $T$  步的  $V$  值函数的综合评估,即相对于即时回报  $R(s_t)$  而言,这部分是  $t$  后可能的回报(强化学习中也被称为未来回报),它需要通过  $V$  值的迭代计算来完成。

### 1.4 动作值函数 $Q(s, a)$

在强化学习中,状态的转移是通过执行动作来完成的。一个状态下实施了某种动作就到达一个新状态。这在机器人系统或者棋牌对弈系统中得到充分体现。例如,在围棋对弈系统中,每落一个棋子就意味着棋局(状态)发生改变,但是这种改变需要持续进行评价。同样,在机器人路线规划中,机器人每前进一步意味着新的状态产生,但是这并不意味着接近目标点,所以每个动作引发的状态更新都需要进行评价并累计到之前的奖励回报中。

假设一个环境用状态集  $S$  和动作集  $A$  来描述。给定  $t$  时刻的当前状态  $s \in S$ , 动作  $a \in A$  是  $s$  状态下可以执行的一个候选动作,则  $\pi$  策略下的关于状态  $s$  和动作集  $a$  的回报可以用动作值函数来估计。公式(5)给出了对应的表达式:

$$Q_{\pi}(s_t, a) \leftarrow R(s_t) + \sum_{k=1}^{T-1} (\gamma^k \times \sum_{b \in A(s_{t+k})} Q_{\pi}(s_{t+k}, b)). \quad (5)$$

其中:  $R(s_t)$  和  $\gamma$  分别是  $t$  时刻的立即回报和折扣因子;  $A(*)$  是状态  $*$  可能采取的下动作集合,整个的后面一项就是  $t$  时刻的未来  $Q$  值的累计估计。

除了经典的上述 3 种算法外,还有在此 3 种算法基础上大量改进的强化学习算法,比如在 Q-Learning 中加入多个智能体,将 TD 与神经网络相结合以及将启发函数与 Sarsa 相结合等等,这些算法都在实验中表现出了优异的成绩。

## 2 Q-Learning 算法分析

### 2.1 基本原理

Q-Learning 是强化学习三种最流行的算法之一,是基于  $Q$  值迭代的无模型算法。如前所述,给定某一时刻的状态  $s_i$  和准备采取的动作  $a_i$ ,  $Q(s_i, a_i)$  就是在该时刻的状态  $s_i$  下采取动作  $a_i$  获得回报

的估计值。理论上讲,当探索了一个给定时刻的状态  $s_i$  和所有可能动作  $A(s_i)$  后,就可以根据环境的反馈回报信息选取一个最优动作进入下一次状态  $s_{i+1}$ ;如此反复直到终点或者人为终止,一次迭代过程结束。当然,一个状态的  $Q$  值也是随着探索的不断推进在不断更新中,直到所有状态的  $Q$  值相对稳定为止。

$Q$ -Learning 算法的特点是根据潜在的状态与动作来构建一张二维表(被称为  $Q$ -table)来存储  $Q$  值,然后通过查表方式获得  $Q$  值来寻找最优的动作。该方法具有简单直接的特点,在环境大小适中的应用场景中(如简单棋牌对弈等),已经证明非常有效。

$Q$ -table 的数据结构简单易用,表 1 给出了一个 4 个状态、2 个动作的结构示意。

表 1  $Q$ -Table 结构示意  
Table 1  $Q$ -Table schematic

$Q$ -Table	$a_1$	$a_2$
$s_1$	$Q(s_1, a_1)$	$Q(s_1, a_2)$
$s_2$	$Q(s_2, a_1)$	$Q(s_2, a_2)$
$s_3$	$Q(s_3, a_1)$	$Q(s_3, a_2)$
$s_4$	$Q(s_4, a_1)$	$Q(s_4, a_2)$

基于  $Q$ -table,  $Q$ -Learning 算法主要使用公式(6)来更新  $Q$  值:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \times ((R(s') + \gamma \times \max_{a' \in A(s')} \{Q(s', a') - Q(s, a)\})). \quad (6)$$

其中: $s$  和  $s'$  分别是当前状态和下一个状态, $a$  则是使  $s$  到  $s'$  的有效动作,而  $A(s')$  则是下状态  $s'$  可能采取的候选动作; $\alpha \in [0, 1]$  被称为为学习率,用于调节学习过程中的可能误差; $\gamma$  为折扣因子。

依据公式(6),  $Q$ -Learning 算法在一个特定状态下将贪婪地对所有可能的路径进行探索,每前进一步都是在寻找当前状态下的局部最优解。

为了准确刻画改进的算法,本文采用的主要符号及其意义见表 2。

表 2 符号表  
Table 2 Symbols in this paper

符号	意义
$S$	环境的状态集
$s$	一个状态
$A$	智能体的动作集
$a$	一个动作
$A(s)$	$s$ 状态下所有候选动作集
$s(a)$	$s$ 状态下采用 $a$ 动作得到的下状态
$R(s)$	$s$ 状态下环境反馈的回报值
$V(s)$	$s$ 状态下 $V$ 值
$Q(s, a)$	$s$ 状态下采用 $a$ 动作生成的 $Q$ 值

算法 1 给出了  $Q$ -Learning 算法的伪码描述。

Input: learning rate  $\alpha$ , discount factor  $\gamma$ , greedy factor  $\epsilon$ , reward matrix  $R(|S|, |A|)$ , maximum forward steps  $maxStep$ , maximum number of  $Q$ -table modifications  $maxIter$ .  
Output:  $Q$ -table  $Q(|S|, |A|)$ .

Process:

Initialize  $Q$ -table: 所有  $Q(s, a)$  元素为 0

iter  $\leftarrow$  0;

Repeat // 循环更新  $Q$  表

$s \leftarrow$  start; 开始状态为 start

step  $\leftarrow$  0;

Repeat // 循环找终点 terminal:

For (each action  $a \in A(s)$ )

call  $\epsilon$ -greedy obtain  $a' \in A(s)$ ; // 用贪婪策略前探

$Q(s, a) \leftarrow Q(s, a) + \alpha * (R(s) + \gamma * (Q(s(a), a') - Q(s, a)))$ ;

$s \leftarrow s(a')$ ;

step++;

Until (step  $\geq$  maxStep) or ( $s$  = terminal)

iter  $\leftarrow$  iter++;

Until (iter  $\geq$  maxIter) or ( $Q$ -table no again change)

Return  $Q$ -Table.

值得注意的是,考虑到现有  $Q$ -Learning 算法的描述大多过于简单<sup>[4,8]</sup>,不利于读者阅读和本文后面算法的介绍。算法 1 把整个  $Q$ -Learning 的主要步骤整合到一起,对细节进行了更详细描述。

简单地讲,  $Q$ -Learning 算法除了必要的初始化外,主要是一个双层循环:

1) 在内层循环里要完成一次从起点到终点状态的探索,并对探索过的路径的  $Q$  值进行更新。它的理想出口是每次探索都能到达终点。这种(内循环)正常出口只意味着找到一条可行的路径来完成从起点到终点的工作,但它不一定是最优的,因此还需要通过  $Q$ -Table 的迭代来寻优。当然,当内循环人为终止(超过步骤阈值)时,就说明本次探索失败,需要重新开始探索。此外,在一个状态下寻找下动作采用的是  $\epsilon$ -greedy 策略。

2) 外循环的正常结束条件是  $Q$ -table 不再变化。这意味着经过多次迭代后,已经产生了从起点到终点状态的稳定情况,因而达到了优化的目标,最后的答案也就是从起点到终点的“最大  $Q$  值”形成的路径。当然,当外循环触动“意外的结束条件”时,就说明循环探索已经达到极限,没有必要继续下去了。

## 2.2 改进 $Q$ -Learning 算法

强化学习和监督学习、半监督学习等其他学习方法不同,它需要不断探索环境并获得反馈。目前最常用的探索策略有两个:贪婪策略和 Boltzmann 策略<sup>[4-5]</sup>。

一般地讲, Boltzmann 基本上是随机选择策略,缺乏探索的目的性,效率很难保证,特别是在复杂情况下很难完成。相对来说,贪婪策略增加了探索的目的性,但是很容易陷入局部优化,甚至经常“碰壁”



而被迫终止。目前的算法也都致力于解决该问题,但是仍然存在很大的改进空间。

本文引入动态搜索因子  $\epsilon$ , 尝试改进 Q-Learning 算法。算法 2 给出了改进 Q-Learning 算法  $\epsilon$ -Q-Learning 的伪码描述。

---

Input: learning rate  $\alpha$ , discount factor  $\gamma$ , basic greedy factor  $\epsilon$ , greed incremental  $\theta$ , reward matrix  $R(|S|, |A|)$ , maximum forward steps maxStep, maximum number of Q-table modifications maxIter.

---

Output: The optimal path  $P$ .

---

Process:

Initialize Q-table; 所有  $Q(s, a)$  元素为 0

---

iter  $\leftarrow 0$ ;

Repeat // 循环更新 Q 表

$s \leftarrow \text{start}$ ; 开始状态为 start

step  $\leftarrow 0$ ;

Repeat // 循环找终点 terminal;

For each action  $a \in A(s)$

IF  $\text{rand}() \leq \epsilon$

Then  $a' \leftarrow$  a random forward action from  $s$

Else

call  $\epsilon$ -greedy get  $a' \in A(s)$ ; // 用贪婪策略前探

$Q(s, a) \leftarrow Q(s, a) + \alpha * (R(s) + \gamma * (Q(s(a), a') - Q(s, a)))$

End Do

$s \leftarrow s(a')$ ;

step  $++$ ;

Until (step  $\geq$  maxStep) or ( $s = \text{terminal}$ )

iter  $\leftarrow \text{iter} + 1$ ;

If (step  $\geq$  maxStep)

Then  $\square \leftarrow \square + \theta$

Else

Then  $\square \leftarrow \square - \theta$ ;

Until (iter  $\geq$  maxIter) or (Q-table no again change);

For ( $s = \text{start}$  To terminal) Do

$a \leftarrow \text{argmax}\{Q(s, a) | a \in A(s)\}$

$P \leftarrow P + \langle a \rangle$ ;

Return  $P$ .

---

$\epsilon$ -Q-Learning 主要改变是根据环境的反馈来动态调整贪婪因子  $\epsilon$ 。正如算法 2 中间描述那样,  $\epsilon$ -greedy 策略是通过条件  $\text{rand}() \leq \epsilon$  来决定是随机选择还是选择最大  $Q$  值的下动作。很显然,  $\epsilon$  越大则随机搜索下动作的概率就会增大, 这在一定程度上避免陷入局部优化。

算法 2 引入动态的贪婪因子  $\epsilon$ 。简单地说, 在算法 2 中, 假如在一次从起点到终点的探索失败, 则通过增大  $\epsilon$  来使下一次探索的随机性增大, 以免陷入之前的局部优化困境。反之, 则通过减少  $\epsilon$  来增加目的性。当然, 算法 2 中的基础贪婪因子  $\epsilon$  和增幅  $\theta$  是一个经验值, 也需要根据应用中环境的不断变化进行尝试实验来确定。

### 3 实验结果及分析

实验在 Anaconda 上采用  $36 \times 36$  的迷宫环境

模拟智能体运动进行仿真, 将改进的  $\epsilon$ -Q-Learning 算法和传统的 Q-Learning 算法进行比较性实验。实验中使用的参数见表 3。

表 3 实验参数设置表

Table 3 Main parameter setting for the experiments

参数	数值
学习率 $\alpha$	0.2
折扣因子 $\gamma$	0.99
探索因子 $\epsilon$	0.10~1 (Q-Learning 使用 0.99)
迭代上线 $n$	2 500
回报 $R$	-0.2
目标回报 $R$	6
障碍物或墙壁回报	-1
增幅 $\theta$	0.005

智能体所在位置的坐标  $(x, y)$  对应 Q-table 的一个状态, 智能体的动作选择有 4 种: 北、南、西、东, 分别用 N、S、W、E 来表示, 这样 Q-table 的容量就是  $1\ 296 \times 4$ 。迷宫中还有障碍物, 若进行动作选择后碰到障碍物和迷宫墙壁则智能体保持在原状态, 否则进入下一个状态。智能体的开始位置设置在迷宫环境的  $(0, 35)$  处, 目标位置设置在  $(35, 0)$  处。

根据上述的实验环境和参数设置, 使用上面的算法 1 和 2, 我们进行了损失函数、运行效率和收敛性等方面的跟踪实验。实验结果如图 2-5 所示。在图中圆点表示的是  $\epsilon$ -Q-Learning 算法的值, 线表示的是传统的 Q-Learning 算法的值。

#### 3.1 损失比较

算法的损失函数所反映的是算法所形成的路径与最优路径的相似度。强化学习中已经出现了 7 种损失函数: 平方误差损失函数、绝对误差损失函数、Huber 损失函数、二元交叉熵损失函数、Hinge 损失函数、多分类交叉熵损失函数、KL 散度损失函数。本文采用是绝对误差损失函数, 公式(7)给出了它的计算方法:

$$L = |y - f(x)| / \text{maxIter}. \quad (7)$$

式中:  $y$  表示每次迭代的路径的总回报,  $f(x)$  表示人为设置最优路径的总回报。显然, 损失函数的值越小越好。图 2 给出了两种算法在损失函数值上的变化趋势。

分析图 2,  $\epsilon$ -Q-Learning 算法和 Q-Learning 算法在迭代次数  $n < 60$  时对环境模型的学习都不够深入, 且  $\epsilon$ -Q-Learning 算法也没有很好体现出其调节  $\epsilon$  效果。然而, 但当  $n$  超过 60 以后, 由于  $\epsilon$ -Q-Learning 算法对环境模型的学习越来越深入,  $\epsilon$  也越来越优化,  $\epsilon$ -Q-Learning 算法就显示出了它明显的优越性, 其损失函数值渐渐小于传统的 Q-Learn-

ing 算法,且更快收敛于稳定值。

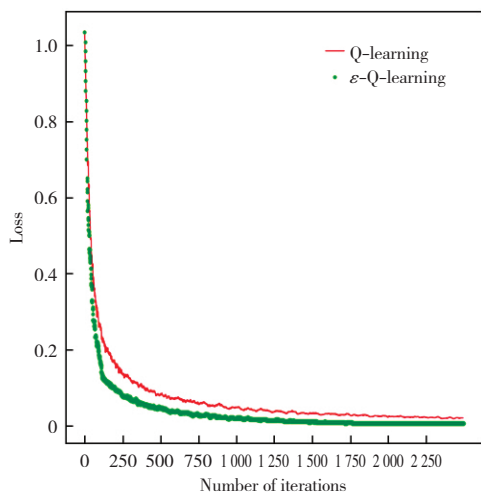


图 2 算法损失函数图  
Fig. 2 Loss function graph

### 3.2 运行效率比较

图 3 给出了两种算法的运行时间比较。

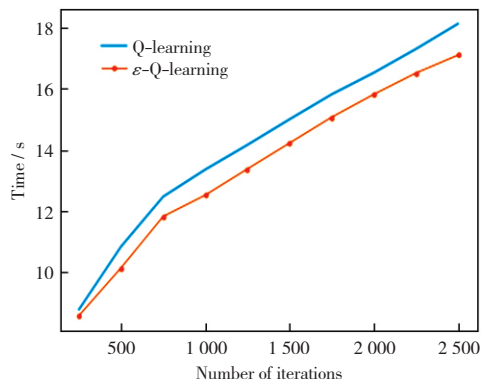


图 3 算法时间图  
Fig. 3 Executive time graph

从图 3 中可以看出,在 250 次迭代之前,ε-Q-Learning 算法的运行时间和 Q-Learning 算法基本没有差距。然而,随着迭代次数的增多,ε-Q-Learning 算法对环境越来越适应,学习效率提升,算法的运行效率更高。

### 3.3 回报函数值比较

迭代过程中总回报函数值的变化反映了算法的收敛情况。图 4 是两个算法的总回报的比较结果。

从图 4 中可以看出在  $n < 60$  时,由于迭代次数太少,ε 的调整效果并不明显,因此 ε-Q-Learning 算法中的回报值与 Q-Learning 算法大致相同。当  $n \geq 60$  时,由于 ε-Q-Learning 开始适应环境,当迭代 500 次后总的回报函数值基本不再变化。因此,从回报函数值的变化,可以推断出 ε-Q-Learning 算法要比 Q-Learning 算法的收敛性要好。

### 3.4 探索步数比较

总的探索步数反映学习算法寻优过程所付出的

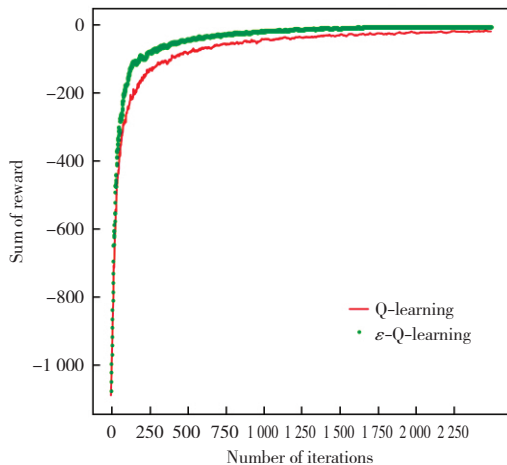


图 4 算法总回报图

Fig. 4 Cumulative rewards graph

代价。一般而言,一个算法使用更少的步数而找到最优解,那么它的代价就小,因而性能也就要好。图 5 给出了两个算法随着迭代次数增加时累计的总步数的变化情况。

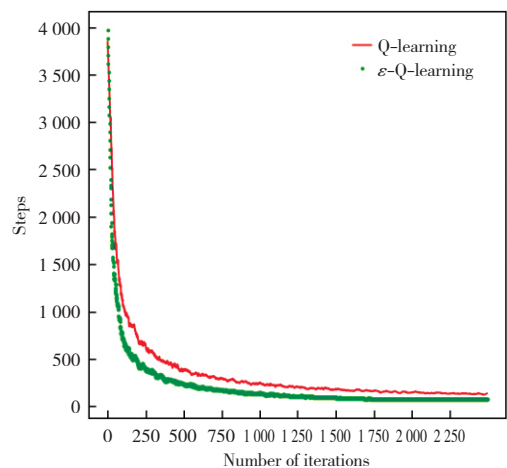


图 5 算法步数图

Fig. 5 Cumulative step graph

从图 5 中可以看出,当  $n < 5$  的时候,ε-Q-Learning 算法的总步数是大于 Q-Learning 算法的,这很有可能是算法随机探索所造成的。当  $5 \leq n < 60$  时,两种算法的总步数大致相当。但是,当  $n \geq 60$  时,ε-Q-Learning 算法步数就逐渐开始小于 Q-Learning 算法的步数,而且随着迭代次数的增多,ε-Q-Learning 算法的总步数基本稳定,收敛也比 Q-Learning 算法要好些。

## 4 总结

传统的 Q-Learning 算法在路径规划中存在收敛速度慢和易陷入局部优化等问题,本文提出了一种改进的 ε-Q-Learning 算法。通过引入动态的探索

因子  $\epsilon$ , 不仅可以有效地提高算法的搜索效率, 而且也能保证最终路径的优化。 $\epsilon$ -Q-Learning 算法根据每一次迭代的总步数来判断本次探索的有效性, 并通过修改贪心算法的搜索因子  $\epsilon$  来提高下次探索的成功率。特别是, 动态的搜索因子技术可以有效地

避免局部搜索困境, 提高探索的成功率。实验结果表明, 改进的  $\epsilon$ -Q-Learning 算法相比现有的 Q-Learning 算法, 在收敛速度、运行效率和搜索代价等方面都表现出不同程度的优势。

### 参考文献:

- [1] KOBER J, BAGNELL J, PETERS J. Reinforcement learning in robotics: a survey[J]. The International Journal of Robotics Research, 2013, 32(11): 1238-1274.
- [2] MOHANAN M, SALGOANKAR A. A survey of robotic motion planning in dynamic environments[J]. Robotics and Autonomous Systems, 2018, 100: 171-185.
- [3] 陈春林. 基于强化学习的移动机器人自主学习及导航控制[D]. 合肥: 中国科学技术大学, 2006.  
CHEN C L. Autonomous learning and navigation control of mobile robot based on reinforcement learning[D]. Hefei: University of Science and Technology of China, 2006.
- [4] SUTTON R S, BARTO A G. Reinforcement learning: an introduction, 2nd ed[M]. Boston, MA, USA: MIT Press, 2018.
- [5] 高阳, 陈世福, 陆鑫. 强化学习研究综述[J]. 自动化学报, 2004(1): 88-102.  
GAO Y, CHEN S F, LU X. Review of reinforcement learning[J]. Journal of Automation, 2004(1): 88-102.
- [6] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: a survey[J]. Journal of Artificial Intelligence Research, 1996, 4: 237-285.
- [7] 刘忠, 李海红, 刘全. 强化学习算法研究[J]. 计算机工程与设计, 2008, 9(22): 5805-5809.  
LIU Z, LI H H, LIU Q. Reinforcement learning algorithm research[J]. Computer Engineering and Design, 2008, 9(22): 5805-5809.
- [8] SILVER D, MADDISON C J, GUEZ A, et al. Mastering the game of go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.
- [9] 许亚. 基于强化学习的移动机器人路径规划研究[D]. 济南: 山东大学, 2013.  
XU Y. Research on path planning of mobile robot based on reinforcement learning[D]. Jinan: Shandong University, 2013.
- [10] 乔俊飞, 侯占军, 阮晓钢. 基于神经网络的强化学习在避障中的应用[J]. 清华大学学报(自然科学版), 2008, 48(S2): 1747-1750.  
QIAO J F, HOU Z J, YAN X G. Application of reinforcement learning based on neural network in obstacle avoidance[J]. Journal of Tsinghua University(Natural Science Edition), 2008, 48(S2): 1747-1750.
- [11] FAKOOR M, KOSARI A, JAFARZADEH M. Humanoid robot path planning with fuzzy markov decision processes[J]. Journal of Applied Research and Technology, 2016, 14(5): 300-310.
- [12] GRZES M, KUDENKO D. Online learning of shaping rewards in reinforcement learning[J]. Neural Networks, 2012, 32(1): 28-36.
- [13] 窦全胜, 周春光, 徐中宇, 等. 动态优化环境下的群核进化粒子群优化方法[J]. 计算机研究与发展, 2006, 3(1): 89-95.  
DOU Q S, ZHOU C G, XU Z Y, et al. Group kernel evolutionary particle swarm optimization in dynamic optimization environment[J]. Computer Research and Development, 2006, 3(1): 89-95.
- [14] 赫东锋, 孙树栋. 一种在线自学习的移动机器人模糊导航方法[J]. 西安工业大学学报, 2007, 27(4): 325-329.  
HE D F, SUN S D. An Online self-learning fuzzy navigation method for mobile robots[J]. Journal of Xi'an University of Technology, 2007, 27(4): 325-329.
- [15] 郝钊钊, 方舟, 李平. 基于 Q 学习的无人机三维航迹规划算法[J]. 上海交通大学学报, 2012, 46(12): 1931-1935.  
HAO Z Z, FANG Z, LI P. Three-dimensional track planning algorithm of UAV based on Q-learning[J]. Journal of Shanghai Jiaotong University, 2012, 46(12): 1931-1935.
- [16] ROOZEGAR M. XCS-based reinforcement learning algorithm for motion planning of a spherical mobile robot[J]. Applied Intelligence, 45(3): 736-746.
- [17] MNH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
- [18] 刘智斌, 曾晓勤, 刘惠义, 等. 基于 BP 神经网络的双层启发式强化学习方法[J]. 计算机研究与发展, 2015, 52(3): 579-587.  
LIU Z B, ZENG X Q, LIU H Y, et al. A two-layer heuristic reinforcement learning method based on BP neural network[J]. Computer Research and Development, 2015, 52(3): 579-587.
- [19] SANTOS M, MARTÍN H J A, LÓPEZ V, et al. Dyna-H: a heuristic planning reinforcement learning algorithm applied to role-playing game strategy decision systems[J]. Knowledge-Based Systems, 2010, 23(4): 541-550.

(编辑: 贾丽红)