

强化学习中基于权重梯度下降的函数逼近方法

秦晓燕¹, 刘禹含², 徐云龙³, 李斌⁴

(1. 苏州高博软件技术职业学院信息与软件学院, 江苏 苏州 215163; 2. 滑铁卢大学, 安大略 滑铁卢 N2L3G4;
3. 苏州大学应用技术学院, 江苏 苏州 215325; 4. 苏州大学计算机科学与技术学院, 江苏 苏州 215325)

摘要: 函数逼近法 (function approximation) 是强化学习领域中的一个研究热点, 可以有效处理强化学习中大规模、连续状态和动作空间的问题。基于梯度下降 (gradient descent) 的函数逼近方法虽然是强化学习中使用最广泛的方法之一, 但该算法对步长参数的要求较高, 取值不当易产生收敛速度慢、收敛不稳定甚至发散的情况。针对这类问题, 通过围绕基于函数逼近的 TD (TD, temporal difference) 算法, 在最小二乘方法和梯度下降方法的基础上对权重的更新方法进行了改进, 利用最小二乘方法处理值函数求解权重值, 并结合时序差分 and 梯度下降的思想求出权重之间的误差, 并利用该误差直接更新权重, 从而提出一种权重梯度下降 (WGD, weight gradient descent) 方法。该方法以全新的方式更新权重, 有效降低算法对计算资源的消耗, 并且可以有效地对其他基于梯度下降的函数逼近算法进行改进, 广泛应用于诸多基于梯度下降的强化学习算法。实验表明, WGD 方法能够在更广泛的空间中调整参数, 可以有效降低算法发散的可能性, 在保证算法拥有良好收敛效果的同时, 提高算法的收敛速度。

关键词: 函数逼近; 强化学习; 梯度下降; 最小二乘; 权重梯度下降

中图分类号: TP18

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023050

Function approximation method based on weights gradient descent in reinforcement learning

QIN Xiaoyan¹, LIU Yuhan², XU Yunlong³, LI Bin⁴

1. School of Information and Software, Global Institute of Software Technology, Suzhou 215163, China
2. University of Waterloo, Waterloo, N2L3G4, Canada

收稿日期: 2023-01-06; **修回日期:** 2023-05-30

通信作者: 秦晓燕, 284562303@qq.com

基金项目: 国家自然科学基金 (61772355, 61702055, 61876217, 62176175); 江苏省高等学校自然科学研究重大项目 (18KJA520011, 17KJA520004); 苏州市应用基础研究计划工业部分 (SYG201422); 江苏省高职院校教师专业带头人高端研修项目 (2021GRFX052); 江苏高校优势学科建设工程资助项目; 江苏省职业教育软件技术“双师型”名师工作室资助项目

Foundation Items: The National Natural Science Foundation of China (61772355, 61702055, 61876217, 62176175), Jiangsu Province Natural Science Research University Major Projects (18KJA520011, 17KJA520004), Suzhou Industrial Application of Basic Research Program Part (SYG201422), Jiangsu Province High End Research and Training Project for Professional Leaders of Teachers in Vocational Colleges (2021GRFX052), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Project Funded by Jiangsu Province Vocational Education the Double-Qualified of Teaching Studio in Software Technology

引用格式: 秦晓燕, 刘禹含, 徐云龙, 等. 强化学习中基于权重梯度下降的函数逼近方法[J]. 网络与信息安全学报, 2023, 9(4): 16-28.

Citation Format: QIN X Y, LIU Y H, XU Y L, et al. Function approximation method based on weights gradient descent in reinforcement learning [J]. Chinese Journal of Network and Information Security, 2023, 9(4): 16-28.

3. Applied Technology College, Soochow University, Suzhou 215325, China

4. School of Computer Science and Technology, Soochow University, Suzhou 215325, China

Abstract: Function approximation has gained significant attention in reinforcement learning research as it effectively addresses problems with large-scale, continuous state, and action space. Although the function approximation algorithm based on gradient descent method is one of the most widely used methods in reinforcement learning, it requires careful tuning of the step size parameter as an inappropriate value can lead to slow convergence, unstable convergence, or even divergence. To address these issues, an improvement was made around the temporal-difference (TD) algorithm based on function approximation. The weight update method was enhanced using both the least squares method and gradient descent, resulting in the proposed weights gradient descent (WGD) method. The least squares were used to calculate the weights, combining the ideas of TD and gradient descent to find the error between the weights. And this error was used to directly update the weights. By this method, the weights were updated in a new manner, effectively reducing the consumption of computing resources by the algorithm enhancing other gradient descent-based function approximation algorithms. The WGD method is widely applicable in various gradient descent-based reinforcement learning algorithms. The results show that WGD method can adjust parameters within a wider space, effectively reducing the possibility of algorithm divergence. Additionally, it achieves better performance while improving the convergence speed of the algorithm.

Keywords: function approximation, reinforcement learning, gradient descent, least-squares, weights gradient descent

0 引言

强化学习是机器学习中可用于解决序列决策问题的方法,该方法通过学习主体与环境之间的交互,并利用所得的数值反馈来对当前策略进行评估与控制,求解出能使累积奖赏最大化的策略^[1]。随着研究的不断深入,强化学习在游戏、多智能控制、自主机器人、自然语言处理、智能推荐等众多领域得到了广泛应用^[2-5]。

在强化学习中解决中等或大规模状态空间问题时,传统基于查询表的算法(如 Q-learning 算法、Sarsa 算法)的学习性能会随着状态或动作规模的增加而逐渐降低。函数逼近算法可以利用近似评估值函数实现从空间到状态的映射^[6],从而有效地处理这类问题,并求解出近似最优策略^[7],如 AC 算法^[8]、TD(λ)算法^[9]等。因此函数逼近方法受到了越来越多的关注。

梯度下降方法是函数逼近法中最常用的方法,但是该方法对步长参数的要求较高,取值不当容易导致算法发散,且算法收敛效果的提升需要以收敛速度为代价。随着对算法的研究以及算法的发展,各种优化手段被不断提出,如引入资格迹^[9]或基于模型学习^[10]来加强算法对状态信息的利用,从而同时提升算法的收敛效果和收敛速度,但是这些操作并未对梯度下降方法进行更改^[11]。

本文的研究重点在于对梯度下降方法进行更改,降低其发散的可能性,并同时提升算法的收敛效果与收敛速度。

本文首先从值函数逼近入手,先利用最小二乘的方式进行操作求解权重,再结合时序差分(TD, temporal difference)与梯度下降的思想求出权重误差,并利用该权重差更新权重,从而提出权重梯度下降(WGD, weight gradient descent)方法。该方法拥有广泛的应用空间,可替换各算法中的梯度下降形式,从而形成新的算法。实验表明, WGD 方法可以运用于异策略函数逼近算法、TD(λ)算法,或者 AC 算法的 critic 部分,并且 WGD 方法拥有更广泛的参数取值范围,可以有效保证改进后算法的收敛性。在提升算法收敛速度的同时,可以保证算法拥有良好的收敛效果。从理论而言, WGD 方法的改进方式可以应用于诸多基于梯度下降方法的函数逼近算法,如策略梯度更新^[12]、深度强化学习中的反向传播更新等^[13-14],可以将这些改进工作作为未来的工作重点。

1 理论基础

1.1 马尔可夫决策过程

图 1 展示了强化学习的示意,其中,主体(Agent)在 t 时刻根据其状态与奖赏信息,采取

动作并达到下一状态,得到新的奖赏。通过 Agent 与环境的不断交互, Agent 可以不断优化当前策略,直到找出一个可以将 Agent 所得到的期望累积奖赏最大化的最优策略^[15]。

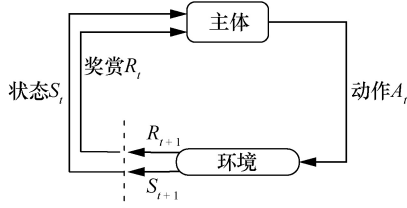


图 1 强化学习示意
Figure 1 Diagram of reinforcement learning

针对强化学习的序列决策问题,可以使用马尔可夫决策过程(MDP, markov decision process)对问题进行建模。通常用五元组 $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ 对 MDP 进行表示。其中, $\mathcal{S}$ 是有界状态空间; $\mathcal{A}$ 是有限动作空间,且 $|\mathcal{A}| < \infty$; p 为状态迁移函数, $p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, $p(s, a, s')$ 表示在状态 $s \in \mathcal{S}$ 下选择动作 $a \in \mathcal{A}$ 并到达后续状态 $s' \in \mathcal{S}$ 的概率; r 为奖赏函数, $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, $r(s, a)$ 表示在状态 s 下采用动作 a 时所得的奖赏,且存在最大奖赏值 $R_{\max}$ 与最小奖赏值 $R_{\min}$ 使得 $R_{\min} \leq r(s, a) \leq R_{\max}$; $\gamma \in (0, 1)$ 为折扣因子,体现出当前所得奖赏的重要性。

强化学习算法根据策略 π 进行决策,并不断优化策略,从而求解出能使收益最大化的策略。 $\pi: \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, $\pi(s, a)$ 表示在状态 s 下采用动作 a 的概率。如果策略 π 为确定策略,则可以将策略 π 定义为从状态空间到动作空间的一个映射,即 $\pi: \mathcal{S} \rightarrow \mathcal{A}$, $a = \pi(s)$ 表示在状态 s 下所选择的动作为 a 。

给定策略 π , 可利用值函数衡量该算法在学习过程中长期表现的优劣。值函数包括状态值函数与状态动作对值函数,可以用未来奖赏的折扣期望和进行求解。在状态 s 下,状态值函数可以定义为 $V(s)$, 当选择的动作为 a 时,可将状态动作对值函数定义为 $Q(s, a)$, 将 t 时刻所在状态记为 s_t , 选择的动作记为 a_t , 所得奖赏记为 r_t : $r_t = r(s_t, a_t)$, 可得如下求解公式。

$$V(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s \right] \quad (1)$$

$$Q(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right] \quad (2)$$

在强化学习中,可以使用 Bellman 公式对这两种值函数进行求解。

$$V(s) = \sum_{a \in \mathcal{A}} \pi(s, a) \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') V(s') \right] \quad (3)$$

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') \sum_{a' \in \mathcal{A}} Q(s', a') \quad (4)$$

若将最优策略 π^* 作为强化学习求解的最终目标,则将最优值函数记为 V^* , 最优动作值函数记为 Q^* 。使用最优 Bellman 公式对其进行如下更新。

$$V^*(s) = \max_{a \in \mathcal{A}} \{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') V^*(s') \} \quad (5)$$

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') \left\{ \max_{a' \in \mathcal{A}} Q^*(s', a') \right\} \quad (6)$$

1.2 函数逼近

当环境中状态规模开始变大时, Agent 无法对所有状态进行访问,也无法为每一个状态设置独立的变量存储其评估值,可采用函数逼近的方式对状态进行近似评估^[16]。函数逼近器为每个状态或状态动作对设置了 d 个基函数(BF, basis function),并对其进行近似特征表示。对状态 s 而言,其特征向量可定义为 $\phi(s)$, 则 $\phi(s) = [\phi_1(s), \phi_2(s), \dots, \phi_d(s)]^T$ 。同理,状态动作对 (s, a) 而言,其特征向量 $\phi(s, a) = [\phi_1(s, a), \phi_2(s, a), \dots, \phi_d(s, a)]^T$ 。与此同时,函数逼近器为所有状态或状态动作对设置了一个 d 维的共享权重 w , 将值函数的近似评估定义为 $\hat{V}(s, w)$ 或 $\hat{Q}(s, a, w)$, 求解公式如下。

$$\hat{V}(s, w) = \phi(s)^T w \quad (7)$$

$$\hat{Q}(s, a, w) = \phi(s, a)^T w \quad (8)$$

在无模型环境中,评估状态动作值函数相对于状态值函数更有效。若无相关说明,本文以下所有公式均对状态动作值函数进行推导说明。在标准的强化学习框架下, Agent 利用采样的方式与环境进行不断地交互,并构成多条有限的马尔可夫链。利用链中数据对权重进行更新时,可以通过梯度下降方法,根据负梯度方向进行迭代更新,从而优化目标函数。

TD 算法是强化学习中最广泛使用的算法之

一，也是构建更复杂方法的基础思想^[17]。在函数逼近方法中，可以将时序差分误差（TD error, temporal-difference error）作为梯度下降的目标函数对权重进行更新。算法通常采用半梯度形式更新算法，其更新公式如下。

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2} \alpha \nabla (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t - \phi(s_t, a_t) \mathbf{w}_t)^2 = \\ &= \mathbf{w}_t + \alpha (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t - \phi(s_t, a_t) \mathbf{w}_t) \phi(s_t, a_t) \end{aligned} \quad (9)$$

2 权重梯度下降方法

本文提出的 WGD 方法从最小二乘方法入手，结合 TD 思想，并采用梯度下降的更新方式更新权重。WGD 方法改变了梯度更新的形式，并可以取得良好的效果。给定策略 π ，在状态 s 下采用动作 a 时，记状态动作对 (s, a) 的真实值为 $Q^\pi(s, a)$ ，将所有状态动作对的真实值组成一个向量，记为 $\mathbf{Q}^\pi$ 。同理，将评估值向量记为 $\hat{\mathbf{Q}}(\mathbf{w})$ 。当算法收敛时，评估值与真实值的误差平方和依旧满足该策略分布下的最小值。

$$\begin{aligned} \mathbf{w} &= \arg \min_{\mathbf{w} \in \mathbb{R}^d} \left\| \mathbf{Q}^\pi - \hat{\mathbf{Q}}(\mathbf{w}) \right\|_\mu^2 = \\ &= \arg \min_{\mathbf{w} \in \mathbb{R}^d} \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \mu(s, a) \left(Q^\pi(s, a) - \hat{Q}(s, a, \mathbf{w}) \right)^2 \end{aligned} \quad (10)$$

其中， μ 为状态分布函数，表示对某一状态或状态动作对的关注程度，当策略 π 稳定时，该分布也随之稳定。

2.1 公式推导

将所有关于状态动作对的特征向量构成特征矩阵标记为 Φ ，关于状态动作对的奖赏组成奖赏矩阵标记为 $\mathbf{R}$ 。令 $\mathbf{D}$ 为与 μ 相关的对角矩阵。在强化学习中，算法的目标是使 $\Phi \mathbf{w} = \mathbf{Q}^\pi$ 。可以利用最小二乘方法的几何意义，找出真实值在评估值函数上的投影，从而直接求解出此时的权重。公式推导如下。

$$\Phi \mathbf{w} = \Phi (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{D} \mathbf{Q}^\pi \quad (11)$$

$$\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{D} \mathbf{Q}^\pi \quad (12)$$

将状态动作对的迁移函数定义为 $f: f(s, a, s', a') = p(s, a, s') \pi(s', a')$ ，取 $\mathbf{F}$ 为所有状态动作对关于函数 f 的迁移矩阵，Bellman 操

作为 $\mathcal{T}: \mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) = \mathbf{R} + \gamma \mathbf{F} \hat{\mathbf{Q}}(\mathbf{w})$ 。在学习过程中，可将 $\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w})$ 作为评估值的更新目标，从而使 $\hat{\mathbf{Q}}(\mathbf{w})$ 不断接近于 $\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w})$ ，权重 $\mathbf{w}$ 不断地接近该策略下的真实权重值，推导公式如下。

$$\mathbf{w} = (\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D} \mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) \quad (13)$$

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbb{E} \left[\left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \right] \\ &\mathbb{E} \left[\phi(s_t, a_t) \mathcal{T} \hat{Q}(s_t, a_t, \mathbf{w}_t) \right] = \\ &\mathbb{E} \left[\left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \right] \\ &\mathbb{E} \left[\phi(s_t, a_t) (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t) \right] \end{aligned} \quad (14)$$

式(14)将算法转为了期望形式，基于期望计算虽然可以加快算法的收敛速度，但前期算法的评估值并不准确。在后期的计算中，这些错误值会一直影响着算法的收敛结果。因此算法难以取得令人满意的结果。本文借用 GTD2 算法的思想对算法的值函数投影误差进行操作，并求出该值误差转移下的权重值误差 δ_w 。权重值误差 δ_w 的求解公式如下。

$$\Phi \delta_w = \Phi (\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D} (\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) - \hat{\mathbf{Q}}(\mathbf{w})) \quad (15)$$

$$\delta_w = (\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D} (\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) - \hat{\mathbf{Q}}(\mathbf{w})) \quad (16)$$

在式(16)中，特征是状态动作对的固有属性，人为定义之后便不再改变。状态分布只与当前所走过的路径相关，不需要通过算法进行学习。本文采用半梯度下降方法对权重误差进行求导，将求导的目标关注于当前状态，并将 $(\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D}$ 作为一个 $d \times |\mathcal{S}|$ 维的常数矩阵进行处理。最终所得求导公式如下。

$$\begin{aligned} \frac{1}{2} \nabla (\delta_w)^2 &= \frac{1}{2} \nabla \left((\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D} (\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) - \hat{\mathbf{Q}}(\mathbf{w})) \right)^2 = \\ &\frac{1}{2} (\Phi^\top \mathbf{D} \Phi)^{-1} \Phi^\top \mathbf{D} * \nabla (\mathcal{T} \hat{\mathbf{Q}}(\mathbf{w}) - \hat{\mathbf{Q}}(\mathbf{w}))^2 = \\ &\frac{1}{2} \sum_{a \in \mathcal{A}, s \in \mathcal{S}} \left(\mu(s, a) \phi(s, a) \phi(s, a)^\top \right)^{-1} \\ &\sum_{a \in \mathcal{A}, s \in \mathcal{S}} \mu(s, a) \phi(s, a) * \nabla (\delta_q(s, a))^2 = \\ &\frac{1}{2} \mathbb{E} \left[\left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \right] \mathbb{E} \left[\phi(s_t, a_t) * \nabla (\delta_q(s_t, a_t))^2 \right] \end{aligned} \quad (17)$$

其中, $*$ 为矩阵中元素的位乘符号, 即 $\mathbf{a} * \mathbf{b} = [a_1 b_1, a_2 b_2, \dots, a_d b_d]$ 。通常情况下, 为了保证算法评估值的准确性, 在对算法进行特征提取时, 尽可能地保证各状态特征向量之间的线性无关性^[17]。因此可将式(17)中的两个期望合并为如下形式。

$$\begin{aligned} \frac{1}{2} \nabla (\delta_w)^2 = \\ \frac{1}{2} \mathbb{E} \left[\left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \phi(s_t, a_t) * \nabla (\delta_q(s_t, a_t))^2 \right] \end{aligned} \quad (18)$$

在采样过程中的每一步, 均可以利用式(18)对权重进行更新, 并称这种更新方法为 WGD 方法。该式利用第一次的样本数据求解误差值, 放弃了二次采样等一些复杂的操作, 可以有效降低历史错误样本数据的影响。而且, 由半梯度下降方法可知, 算法只需要计算 $\nabla (\delta_q(s_t, a_t))^2$ 便可以有效地更新权重并求出权重值。因此, 当样本数据充足时, 式(18)中的期望值依旧为零向量。式中 $(\phi(s_t, a_t) \phi(s_t, a_t)^\top)^{-1} \phi(s_t, a_t)$ 对算法的影响虽然弱, 但是却可以有效地利用当前样本的状态信息。式中的评估值误差 δ_q 为 TD 误差, 权重的具体更新公式如下。

$$\begin{aligned} \mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \frac{1}{2} \nabla (\delta_w)^2 = \\ \mathbf{w}_t - \alpha \left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \phi(s_t, a_t) \\ \frac{1}{2} * \nabla (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t - \phi(s_t, a_t) \mathbf{w}_t)^2 = \\ \mathbf{w}_t + \alpha \left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \phi(s_t, a_t) \cdot \\ (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t - \phi(s_t, a_t) \mathbf{w}_t) * \phi(s_t, a_t) \end{aligned} \quad (19)$$

其中, 更新公式中的 TD 误差由 Sarsa 算法控制, 因此可将该算法称为 WGD-Sarsa 算法。继续对式(19)进行优化, 将内部的矩阵计算转换为向量计算, 从而降低算法的计算量。矩阵计算的优化过程如下。

$$\begin{aligned} \left(\phi(s_t, a_t) \phi(s_t, a_t)^\top \right)^{-1} \phi(s_t, a_t) = \\ \frac{[\phi(s_t, a_t), \phi_2(s_t, a_t), \dots, \phi_d(s_t, a_t)]^\top}{\sum_{i=1}^d (\phi_i(s_t, a_t))^2} = \frac{\phi(s_t, a_t)}{\sum_{i=1}^d (\phi_i(s_t, a_t))^2} \end{aligned} \quad (20)$$

$$\text{令 } \beta(s_t, a_t) = \frac{1}{\sum_{i=1}^d (\phi_i(s_t, a_t))^2}, \text{ 此时, 对任意状}$$

态 s 而言, 均存在 $\phi(s, a)^\top \beta(s, a) \phi(s, a) \approx 1$ 。最后将式(20)代入式(19)中, 最终所得公式如下。

$$\begin{aligned} \mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \beta(s_t, a_t) \phi(s_t, a_t) \\ (r_t + \gamma \phi(s_{t+1}, a_{t+1}) \mathbf{w}_t - \phi(s_t, a_t) \mathbf{w}_t) * \phi(s_t, a_t) \end{aligned} \quad (21)$$

在公式推导过程中可知, WGD 方法在更新权重时, 利用投影矩阵求出权重误差, 并使用半梯度下降方法求解权重。因此该方法完全放弃了最小二乘方法的优势, 无法充分利用全部的样本数据。同时, 该方法对特征提取的要求较高, 为了保证算法拥有较好的学习性能, 对于不相关的状态, 需要尽可能地保证各状态特征之间的线性无关性, 因此需要选择合适的特征提取方法。但是该方法对计算资源的消耗较低, 而且在更新时可以有效利用当前样本的特征信息调整权重更新的幅度。

2.2 权重梯度下降方法分析

通过上述分析, 可给出如下关于 WGD-Sarsa 算法的流程。

算法 1 WGD-Sarsa 算法

- 1) 输入: 步长参数 α , 基函数 ϕ , 折扣因子 γ , 探索因子 ε , 函数 β
- 2) 输出: 权重向量 $\mathbf{w}$
- 3) 初始化设置: 时间 $t = 0$
- 4) Repeat 1:
- 5) 初始化环境, 状态 s , 动作 a
- 6) Repeat 2:
- 7) 执行动作 a , 得到奖赏 r 和状态 s'
- 8) 如果 s' 为终止状态, 结束 Repeat 2
- 9) $t = t + 1$
- 10) 根据 ε 选择动作 a'
- 11) $\mathbf{w} = \mathbf{w} + \alpha \beta(s, a) \phi(s, a) (r + \gamma \phi(s', a') \mathbf{w} - \phi(s, a) \mathbf{w}) * \phi(s, a)$
- 12) $s = s', a = a'$
- 13) 当算法达到收敛要求, 结束 Repeat 1
- 14) Return $\mathbf{w}$

在算法流程中, 步骤 11) 为该算法的主要更新公式。与原本的梯度下降方法不同, WGD 方法先将评估值以最小二乘的方式进行处理, 利用

Sarsa 算法控制 TD 误差, 并在此基础上利用梯度下降的方式更新权重, 从而形成 WGD-Sarsa 算法。在实际操作中, 对状态动作的特征提取方法并不唯一, 会产生特征取值偏高或者偏低的情况。而 WGD 方法中, $\beta(s,a)\phi(s,a)$ 也会做出相应的变化, 从而间接调整权重的更新幅度。

同时, 步骤 11) 为关于 WGD 方法在 Sarsa 算法的应用, 如果是针对值函数的更新, 可以将公式转换成如下形式, 记为 WGD-V 算法。

$$\begin{aligned} \mathbf{w} = \mathbf{w} + \alpha \beta(s) \phi(s) \cdot \\ (r + \gamma \phi(s') \mathbf{w} - \phi(s) \mathbf{w}) * \phi(s) \end{aligned} \quad (22)$$

其中, $\beta(s) = \frac{1}{\sum_{i=1}^d (\phi(s)_i)^2}$ 。WGD 方法还拥有很大

的改进空间与广泛的应用范围, 基于梯度下降方法的诸多算法以及对算法的改进方式对于 WGD 方法而言, 同样适用, 且能取得较好的学习效果。

以动作值函数为例, 可以将 WGD 方法应用到异策略函数逼近问题中, 引入重要性采样的方式对算法进行改进, 形成异策略函数逼近算法。假设目标策略为 π , 行为策略为 b , 将重要性采样率定义为 ρ : $\rho(s,a) = \pi(s,a) / b(s,a)$, 可形成如下关于 WGD-Sarsa 算法的异策略函数逼近公式, 并替换原流程中的步骤 11), 更新公式如下。

$$\begin{aligned} \mathbf{w} = \mathbf{w} + \alpha \rho(s,a) \beta(s,a) \phi(s,a) \cdot \\ (r + \gamma \phi(s',a') \mathbf{w} - \phi(s,a) \mathbf{w}) * \phi(s,a) \end{aligned} \quad (23)$$

WGD 方法可以引入向前看或者向后看机制形成新算法。如 Sarsa(λ) 算法一样, 该算法需要设置一个轨迹向量 $\mathbf{z}$ 。轨迹向量 $\mathbf{z}$ 与 $\phi(s,a)$ 有关, 则该算法记为 WGD-Sarsa1(λ) 算法, 与 $\beta(s,a) \phi(s,a) * \phi(s,a)$ 有关, 记为 WGD-Sarsa2(λ) 算法。通过大量的实验证明对比, 两个算法均能提升原算法的学习效果, 且拥有良好的收敛性。WGD-Sarsa1(λ) 更新公式如下。

$$\mathbf{z} = \alpha \lambda \mathbf{z} + \phi(s,a) \quad (24)$$

$$\begin{aligned} \mathbf{w} = \mathbf{w} + \alpha \beta(s,a) \phi(s,a) \cdot \\ (r + \gamma \phi(s',a') \mathbf{w} - \phi(s,a) \mathbf{w}) * \mathbf{z} \end{aligned} \quad (25)$$

WGD-Sarsa2(λ) 更新公式如下。

$$\mathbf{z} = \alpha \lambda \mathbf{z} + \beta(s,a) \phi(s,a) * \phi(s,a) \quad (26)$$

$$\mathbf{w} = \mathbf{w} + \alpha (r + \gamma \phi(s',a') \mathbf{w} - \phi(s,a) \mathbf{w}) \mathbf{z} \quad (27)$$

同时, WGD 方法可以引入 AC 算法中。关于 Critic 权重 $\mathbf{w}$ 的更新可以利用 WGD 方法进行改进, 然而关于 Actor 的权重 θ 的更新相对较复杂, 暂时不使用 WGD 的方式进行改进, 可以将其作为将来的一个研究点。将关于权重 $\mathbf{w}$ 的步长参数记为 α^w , 关于权重 θ 的步长参数记为 α^θ , 得到 WGD-AC 算法, 其更新公式如下。

$$\begin{aligned} \mathbf{w} = \mathbf{w} + \alpha^w \beta(s,a) \phi(s,a) \cdot \\ (r + \gamma \phi(s',a') \mathbf{w} - \phi(s,a) \mathbf{w}) * \phi(s,a) \end{aligned} \quad (28)$$

$$\begin{aligned} \theta = \theta + \alpha^\theta \gamma^t (r + \gamma \phi(s',a') \mathbf{w} - \phi(s,a) \mathbf{w}) \cdot \\ \nabla_\theta \ln \pi(a|s, \theta) \end{aligned} \quad (29)$$

上述算法均是对基础算法进行改进所得到的新算法。理论上, 任何基于梯度下降的算法均可以使用 WGD 方法进行替换并形成更好的算法。因此, 可以将这些改进工作作为未来的重点考虑对象。

2.3 收敛性分析

为证明算法的收敛性, 本文以 WGD-Sarsa 算法为例, 将式(20)转换成如下形式。

$$\begin{aligned} \mathbf{w}_{t+1} = (\mathbf{I} - \alpha \beta(s_t, a_t) \phi(s_t, a_t) * \phi(s_t, a_t) \cdot \\ (\phi(s_t, a_t) - \gamma \phi(s_{t+1}, a_{t+1}))^T) \mathbf{w}_t + \\ \alpha \beta(s_t, a_t) \phi(s_t, a_t) * \phi(s_t, a_t) r_t \end{aligned} \quad (30)$$

其中, $\mathbf{I}$ 为单位矩阵, 令 $\mathbf{w}_{t+1} = (\mathbf{I} - \alpha \mathbf{A}) \mathbf{w}_t + \alpha \beta(s_t, a_t) \phi(s_t, a_t) * \phi(s_t, a_t) r_t$, 对式中的矩阵 $\mathbf{A}$ 进行如下操作。

$$\begin{aligned} \mathbf{A} = \mathbb{E} [& \beta(s_t, a_t) \phi(s_t, a_t) * \phi(s_t, a_t) \cdot \\ & (\phi(s_t, a_t) - \gamma \phi(s_{t+1}, a_{t+1}))^T] = \\ & \sum_s \mu(s, a) \sum_{r, s'} p(s, a, s') \pi(s', a') \beta(s, a) \\ & \phi(s, a) * \phi(s, a) (\phi(s, a) - \gamma \phi(s', a'))^T = \\ & \sum_s \mu(s, a) \sum_{r, s'} f(s, a, s', a') \beta(s, a) \cdot \\ & \phi(s, a) * \phi(s, a) (\phi(s, a) - \gamma \phi(s', a'))^T = \\ & \sum_s \mu(s, a) \beta(s, a) \phi(s, a) * \phi(s, a) \cdot \\ & \left(\phi(s, a) - \gamma \sum_{s', a'} f(s, a, s', a') \phi(s', a') \right)^T = \\ & (\beta * \Phi^T * \Phi^T) \mathbf{D} (\mathbf{I} - \gamma \mathbf{F}) \Phi \end{aligned} \quad (31)$$

其中, β 为所有状态动作对关于 β 函数的列向量。在矩阵 $D(I - \gamma F)$ 中, 其对角线中的元素值为正, 而对角线外的元素均为负值。假设在矩阵 M 中, 向量元素的和操作为 $\mathbf{1}^T$, 则存在如下公式。

$$\mathbf{1}^T M = \left[\sum_i M_{1,i}, \sum_i M_{2,i}, \dots, \sum_i M_{n,i} \right]^T \quad (32)$$

其中, $M_{n,i}$ 为矩阵中 n 行 i 列元素。令 μ 是关于 μ 的所有状态动作对分布的列向量。可以将上述公式中的部分计算进行如下操作。

$$\begin{aligned} \mathbf{1}^T D(I - \gamma F) &= \mu^T (I - \gamma F) = \\ \mu^T - \gamma \mu^T F &= \mu^T - \gamma \mu^T = (1 - \gamma) \mu^T \end{aligned} \quad (33)$$

矩阵 A 的最终化简结果如下。

$$\begin{aligned} A &= (1 - \gamma)(\beta * \Phi^T * \Phi^T * \mu^T) \Phi = \\ (1 - \gamma)(\beta * \Phi^T * \Phi^T) D \Phi \end{aligned} \quad (34)$$

由上述公式推导可知, 如果 α 取值足够小, 就可以保证 αA 矩阵中所有元素均无限接近于 0, 从而有效保证算法的收敛性。同时, 算法收敛的结果和状态动作对的概率分布与 D 相关, 在采用样本更新时, 算法进行不断探索, 对所有的状态动作对进行访问, 因此算法所得的实际分布会不断逼近该策略下的真实分布, 此时求解出的权重即最优 TD 权重。然而, 在实际情况下, 上述条件并不能成立, 但可以保证算法得到权重为当前实际分布下的最优权重。

3 实验

为了验证 WGD 方法的有效性, 以及该方法在各种算法中的应用能力, 本文以强化学习中的 A-presplit 反例、Baird 反例、Mountain Car 实验与 Cartpole 实验为例对算法进行性能对比。在将 WGD 方法应用到基于梯度下降的各种函数逼近算法中时, A-presplit 反例可以验证算法收敛值的准确性。Baird 反例可以验证算法在异策略问题中的收敛性。Mountain Car 实验可以用于验证基于 WGD 方法在实际问题中的应用情况, 并体现出该方法对算法的改进能力。

3.1 收敛性分析

A-presplit 反例是强化学习中的一个经典问

题, 如图 2 所示, 该实验包含 4 个状态, 状态 A_1 和状态 A_2 的特征表示相同, 可记为状态 A, 且状态 A、B、C 的特征表示均不相同。情节从状态 A 出发, 分别到达状态 B 和 C, 并得到为 0 的奖赏。然后从 B 到达终点得到为 1 的奖赏或从 C 到达终点得到为 1 的奖赏。其中状态 A_1 和状态 A_2 被选择的概率相同, 且折扣因子 $\gamma = 0$ 。Baird 反例用于说明基于 WGD 的异策略算法在函数逼近问题中的收敛性情况。如图 3 所示, 该实验包含 7 个状态, 其中目标策略为 π , 行为策略为 b , 折扣因子为 γ 。在环境中, 所有状态转移立即奖赏均为 0, 由图 3 可知, 采取实线动作均能到达底层状态, 采用虚线动作是等概率地到达上层 6 个状态之一。本文实验以 WGD-V 算法为例对 WGD 方法进行分析说明。

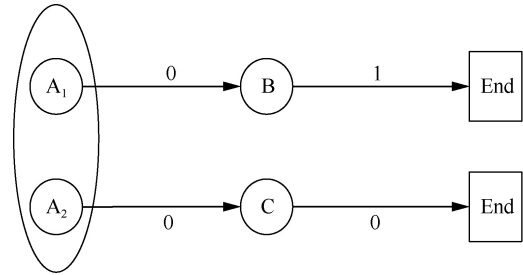


图 2 A-presplit 反例示意
Figure 2 Diagram of A-presplit counter example

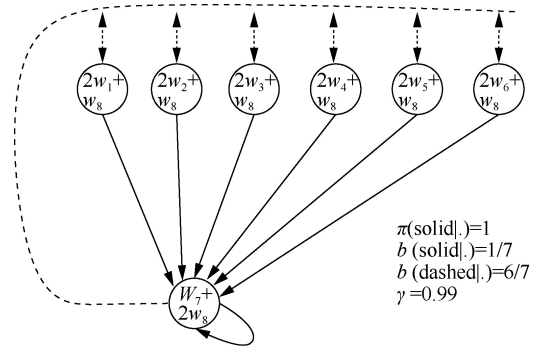


图 3 Baird 反例示意
Figure 3 Diagram of Baird counter example

图 4 为 A-presplit 反例实验中 WGD-V 算法收敛情况, WGD-V 算法在 A-presplit 实验中运行 200 个情节, 并重复运行 100 次。其中步长参数 $\alpha = 0.1$ 。在该实验中, 共有两个情节。在情节 A→B→End 中, 所求得真实值为 A=1, B=1; 在情节 A→C→End 中, 所求得真实值为 A=0, C=0。由此可知 B 与 C 的真实值为 1 和 0, 由于两情节

发生的概率相同, A 的真实值为 0.5。从图 4 可知, A、B 和 C 的评估值均从 1 开始, WGD-V 算法对 A 值的评估可以维持在正确的值上, 对 C 值的评估也在不断地趋向于其真实值。对于 B 值来说, 残差梯度 (RG, residual gradient) 算法在评估时, 得到了一个错误的评估值, 而 WGD-V 算法可以有效地避免这种情况。由此可知, WGD-V 算法虽然改变了梯度下降的更新形式, 但算法依旧可以将权重收敛到正确值。

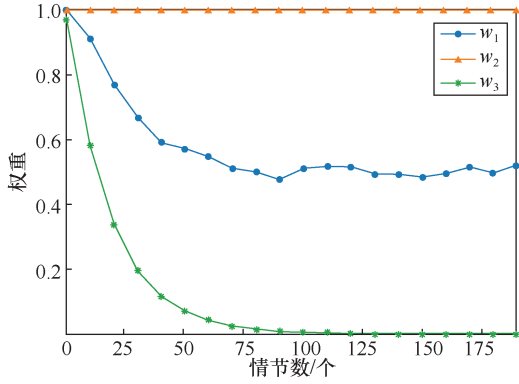


图 4 A-presplit 反例实验中 WGD-V 算法收敛情况
Figure 4 Convergence of the WGD-V algorithm in A-presplit counter example experiment

图 5 为 Baird 反例实验中异策略 WGD-V 算法收敛情况, 该实验为算法运行了 100 个情节, 且重复运行 100 次, 体现了 WGD-V 算法在异策略函数逼近中的应用, 其更新公式如下。

$$\begin{aligned} w = w + \alpha \frac{\pi(s', a')}{b(s', a')} \beta(s) \phi(s) \cdot \\ (r + \gamma \phi(s') w - \phi(s) w) * \phi(s) \end{aligned} \quad (35)$$

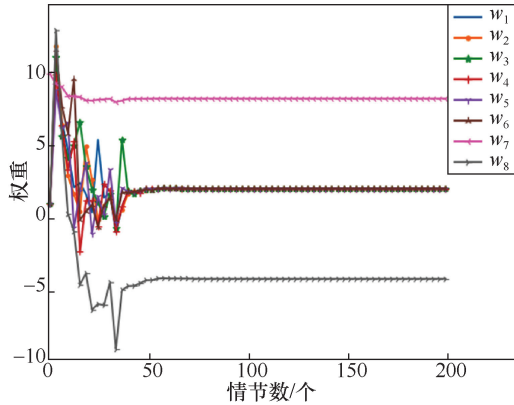


图 5 Baird 反例实验中异策略 WGD-V 算法收敛情况
Figure 5 Convergence of the off-policy WGD-V algorithm in Baird counter example experiment

此时算法的步长参数 $\alpha = 0.15$ 。实验中各状态的真实值均为 0, 权重值 $w = \omega[1, 1, 1, 1, 1, 1, 4, -2]^T$, 其中 ω 为任意不为 0 的实数。从图中可以看出, WGD 方法改变了半梯度方法在异策略函数逼近算法中易发散的特性。它在保障算法收敛性的同时, 能使算法收敛到正确值。

为了更好地体现 WGD 方法对算法的改进能力, 在 Baird 反例中引入均方值误差 (MSVE, mean square value error) 对算法性能进行衡量。图 6 为 Baird 反例实验中不同算法的 MSVE, 体现各算法重复运行 100 次之后所得的平均效果。其中, RG 算法的步长参数 $\alpha = 0.005$ 。该算法以 Bellman 误差为目标求解权重, 在一些特殊的实验中 (如 A-presplit 反例), 该算法会收敛至错误值。但大部分环境下, 该算法可以收敛至正确值, 并且拥有较快的收敛速度^[18]。GTD2-MP 算法由 Ghavamzadeh 等^[19]提出, 是基于 GTD2 算法改进所得, 其参数 $\alpha = 0.01$, $\beta = 0.03$ 。该算法即使不使用二次采用的方式也可以有效提高 GTD2 算法的收敛速度。WGD-V 算法中步长参数 $\alpha = 0.18$ 。这 3 种算法都更改了梯度下降的更新形式, 且此时的参数取值均能使算法取得较快的收敛速度与较低的 MSVE 值。

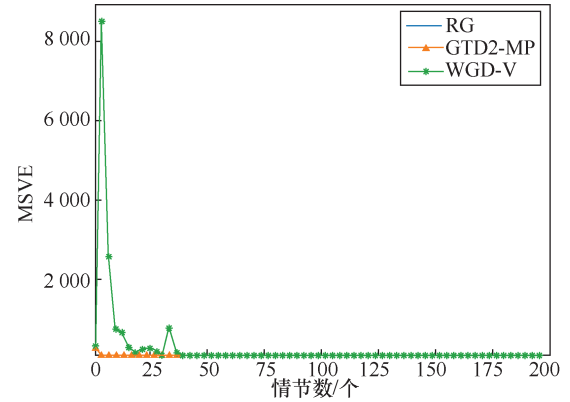


图 6 Baird 反例实验中不同算法的 MSVE
Figure 6 MSVE of different algorithms in Baird counter example experiment

从图 6 可知, 由于 RG 算法的性能与 GTD2-MP 算法相似, 已被覆盖, 而 WGD-V 算法在收敛速度上稍微慢些, 3 种算法最终都收敛到了正确的值, 并将 MSVE 值减小到趋近于 0。

同时, 与图 5 的结果进行参照可知, WGD-V

算法并没有使用二次采样的方式选取样本，在计算上也没有 GTD2-MP 算法那么复杂。因此，在前 50 个情节下，各状态的值变化波动较大，收敛速度上有所欠缺。但这也意味着该算法在前期会得到更多的探索优势，而 GTD2-MP 算法和 RG 算法虽然收敛速度较快，后期每个情节中所运行的情节也基本相同，故它所得到的 MSVE 不会继续降低，而 WGD-V 算法却可以继续对各状态的值进行优化，并得到更低的 MSVE 值。此时，对算法的 MSVE 值进行统计，可得到如下收敛结果：MSVE(WGD-V) $\approx 8.54 \times 10^{-17}$ ，MSVE (GTD2-MP) $\approx 7.56 \times 10^{-4}$ ，MSVE(RG) $\approx 8.62 \times 10^{-15}$ 。由此可知，WGD-V 算法在收敛时，对状态值的评估更为准确。

3.2 Mountain Car 实验与 Cartpole 实验

Mountain Car 实验与 Cartpole 实验是强化学习中经典的连续状态空间案例，可以用于分析基于 WGD 方法中 TD 算法的学习性能，并且体现出 WGD 方法对各种 TD 算法的优化能力。

Mountain Car 实验示例如图 7 所示。

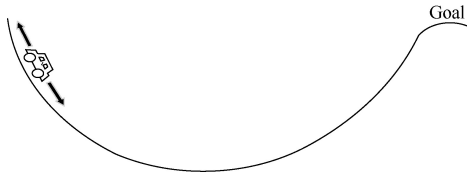


图 7 Mountain Car 实验示例
Figure 7 Diagram of Mountain Car experiment

小车从山坡的左侧出发，通过在两坡间不断徘徊的方式获取足够的动力，从而到达右侧的山顶。实验中，小车的坐标 p ，车速 v 以及动作 a 共同构成了小车的特征。在 t 时刻，小车的 3 个特征之间满足如下关系。

$$p_{t+1} = \text{bound}[p_t + v_t]$$

$$v_{t+1} = \text{bound}[v_t + 0.001a_t - 0.0025\cos(3p_t)]$$

其中， bound 为限界函数，并规定了 $-1.2 \leq p_t \leq 0.5$ ， $-0.07 \leq v_t \leq 0.07$ 。其中小车的动作取值 a_t 为 -1, 0 和 1，分别表示小车向左加速、不加速和向右加速。实验中，小车从坐标为 $p_0 = -0.5$ 的位置出发，起始速度 $v_0 = 0$ 。在行驶过程中，小车得到的奖赏均为 0。当小车到达目标时，一个情节结束且获得值为 1 的立即奖赏。

实验中，本文采用 Tile 编码方法将状态的特征表示从 3 维升至 1 000 维，将连续状态空间转为离散状态空间。Tile 编码中，tiling 数为 8 且 tiling 中每一个 tile 的大小均为 1。

Cartpole 实验示意如图 8 所示，图中的小车可以向左或者向右行驶，小车上的平衡杆会因为小车的移动而向左或者向右摇摆。当平衡杆倾斜一定角度或者小车开出指定范围时，一次实验结束。与 Mountain Car 实验相反，该实验需要小车坚持足够长的时间以保证小车的正常运行。

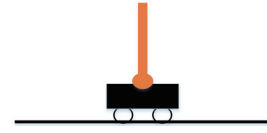


图 8 Cartpole 实验示意
Figure 8 Diagram of Cartpole experiment

实验中，小车的位置 cp ，速度 cv ，平衡杆倾斜角度的正弦值 pp ，平衡杆转动的角速度 pv 和小车所选择的动作 a 共同构成小车的特征。其中，当小车选择动作为向右加速、不加速和向左加速时，动作 a 的值分别为 1、0 和 -1。为了使实验中的算法取得更好的学习性能，本文利用 Tile 编码将连续的 5 维特征转为离散的 2 000 维特征，其中，tiling 数为 8 且 tiling 中每一个 tile 的大小均为 1。当一个情节结束时，小车得到的奖赏为 -100。反之，则奖赏函数 r 与小车各特征之间的关系如下。

$$r = \frac{4.8 - \text{abs}(cp)}{4.8} + \frac{8 - \text{abs}(cv)}{8} + \frac{2\sin\left(\frac{\pi}{15}\right) - \text{abs}(pp)}{2\sin\left(\frac{\pi}{15}\right)} + \frac{10.6 - \text{abs}(pv)}{10.6} \quad (36)$$

最后，将基于 WGD 方法、最小二乘方法以及梯度下降方法的各算法（如 LS-Sarsa、WGD-Sarsa 等）应用于该实验，并分析各算法在实验中算法的学习性能。其中，图 9~图 14 为各算法在两实验中重复运行 100 次后所得的平均步数。Mountain Car 实验中，算法的探索率均为 $\varepsilon = 0.01$ 。而 Cartpole 实验中算法的探索率会随着时间的增加而逐渐降低，其公式满足 $\varepsilon = 0.9^t$ 。

图9和图10用于分析WGD方法对基于Sarsa算法的改进能力。图9为Mountain Car实验中基于Sarsa算法运行500个情节的平均效果。其中，Sarsa算法的步长参数 $\alpha = 0.1$ ，WGD-Sarsa算法的步长参数 $\alpha = 0.8$ 。图10为Cartpole实验中各算法运行500个情节的平均效果。其中，Sarsa算法中 $\alpha = 0.1$ ，WGD-Sarsa算法中 $\alpha = 0.5$ 。在两个实验中，LS-Sarsa算法虽然可以快速收敛，但是算法的收敛效果却并非最好，需要后期增加探索来优化算法的学习性能。Sarsa算法的收敛值优于LS-Sarsa算法，但算法的收敛速度却很慢。受步长参数 α 的影响，Sarsa算法无法保证收敛速度与收敛效果同时达到最优。无论是Mountain Car实验还是Cartpole实验，WGD-Sarsa算法的收敛值与收敛效果都优于Sarsa算法，且该算法的收敛效果是所有算法中最好的。

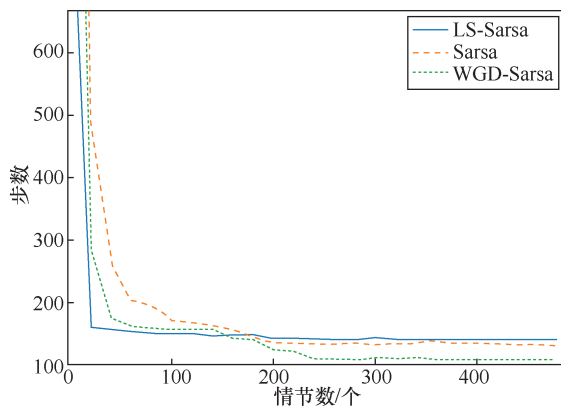


图9 Mountain Car 实验中基于 Sarsa 算法的实验对比
Figure 9 Comparison of algorithm experiments based on Sarsa in Mountain Car experiment

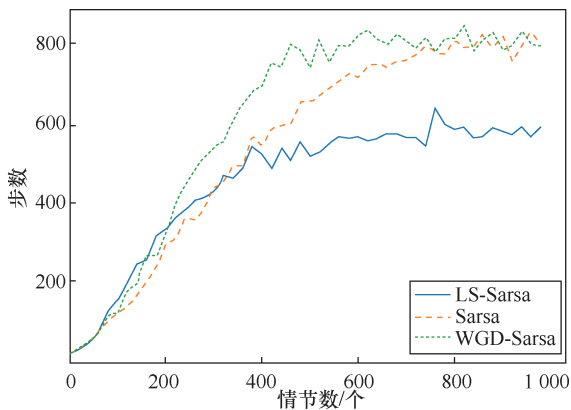


图10 Cartpole 实验中基于 Sarsa 算法的实验对比
Figure 10 Comparison of algorithm experiments based on Sarsa in Cartpole experiment

图11和图12用于对比分析WGD方法在资格迹中的应用效果。图11为Mountain Car实验中基于资格迹算法运行500个情节的平均效果，其中LS-Sarsa(λ)算法中 $\lambda = 0.1$ ，Sarsa(λ)算法中 $\alpha = 0.1$ ， $\lambda = 0.2$ ，WGD-Sarsa1(λ)算法中 $\alpha = 0.3$ ， $\lambda = 0.1$ ，WGD-Sarsa2(λ)算法中 $\alpha = 0.8$ ， $\lambda = 0.6$ 。由图可知，LS-Sarsa(λ)算法虽然收敛速度最快，但后期的收敛值反而最高。Sarsa(λ)算法在引入资格迹方法后，其性能得到明显提升，拥有较低的收敛值。WGD-Sarsa1(λ)算法对资格迹方法的使用较为简单，虽然它的收敛速度最慢，但是它在后期收敛时的效果优于上述两种算法。WGD-Sarsa2(λ)算法对资格迹的处理相比WGD-Sarsa1(λ)算法而言更加复杂。该算法拥有较快的收敛速度，而且算法的收敛值是所有算法中最好的。图12为Cartpole实验中基于资格迹算法运行500个情节的平均效果，其中，LS-Sarsa(λ)算法中 $\lambda = 0.7$ ，Sarsa(λ)算法中 $\alpha = 0.1$ ， $\lambda = 0.7$ ，WGD-Sarsa1(λ)算法中 $\alpha = 0.5$ ， $\lambda = 0.4$ ，WGD-Sarsa2(λ)算法中 $\alpha = 0.3$ ， $\lambda = 0.9$ 。对比可以看出，LS-Sarsa(λ)算法依旧是所有算法中效果最差的。WGD-Sarsa1(λ)算法对资格迹方法的使用并不完善，因此算法的收敛效果差于Sarsa(λ)算法。WGD-Sarsa2(λ)算法不仅拥有更快的收敛速度，并且算法的收敛效果也是所有算法中最好的。由此可知，在基于资格迹的TD算法中，WGD方法可以有效地提升算法的收敛速度与收敛效果。

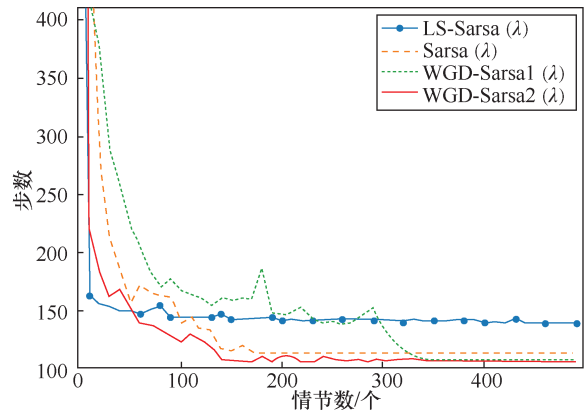


图11 Mountain Car 实验中基于资格迹算法的实验对比
Figure 11 Comparison of algorithm experiments based on eligibility traces in Mountain Car experiment

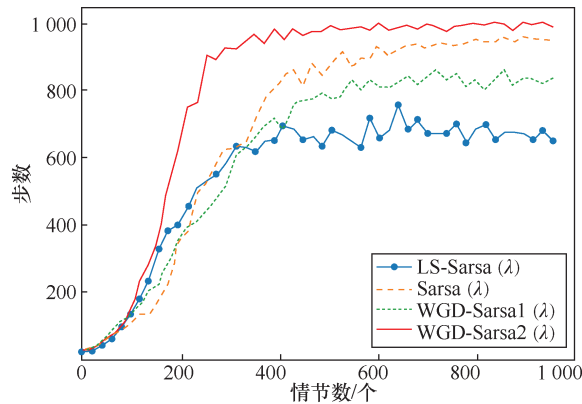


图 12 Cartpole 实验中基于资格迹算法的实验对比
Figure 12 Comparison of algorithm experiments based on eligibility traces in Cartpole experiment

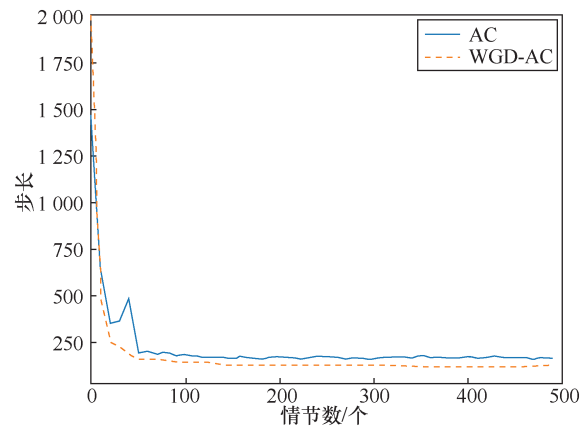


图 13 Mountain Car 中基于 AC 算法的实验对比
Figure 13 Comparison of algorithm experiments based on AC in Mountain Car experiment

图 13 和图 14 用于对比分析 WGD 方法在基于 AC 算法中的应用。图 13 为 Mountain Car 实验中基于 AC 算法运行 500 个情节的平均效果, 其中, AC 算法中 $\alpha^w = 0.1$, $\alpha^\theta = 0.01$, WGD-AC 算法中 $\alpha^w = 0.9$, $\alpha^\theta = 0.01$ 。图 14 中可见 Cartpole 实验中基于 AC 算法运行 500 个情节的平均效果, 其中, AC 算法中 $\alpha^w = 0.1$, $\alpha^\theta = 0.04$, WGD-AC 算法中 $\alpha^w = 0.7$, $\alpha^\theta = 0.5$ 。对比可以看出, WGD-AC 算法无论是收敛速度还是收敛效果都有明显的优势。AC 算法中行动者所选择动作的好坏与评论家的评估能力密切相关。引入 WGD 方法的 AC 算法提升了评论家的评估性能, 从而提高了行动者的动作选择能力, 因此算法的学习性能得到极大改善。

为了更好地分析 WGD 方法对 TD 算法的改进能力, 将上述实验中所有算法的数据进行整合,

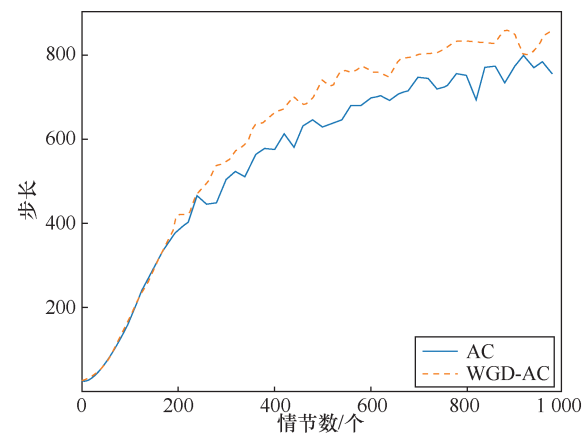


图 14 Cartpole 中基于 AC 算法的实验对比
Figure 14 Comparison of algorithm experiments based on AC in Cartpole experiment

统计出算法在 100 轮实验循环下, 最后 50 个情节的平均步数。表 1 和表 2 分别为 Mountain Car 和

表 1 Mountain Car 实验中不同步长参数取值下各算法的平均收敛效果数据统计
Table 1 Statistics on the average convergence performance of each algorithm under different step size parameter values in the Mountain Car experiment

算法	$(\alpha^w \backslash \alpha^\theta)$								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sarsa	133.9	140.7	X	X	X	X	X	X	X
WGD-Sarsa	165.3	151.6	115.6	118.7	112.3	111.7	116.8	109.4	120.7
Sarsa(λ), $\lambda=0.1$	136.2	145.5	X	X	X	X	X	X	X
Sarsa(λ), $\lambda=0.3$	124.0	X	X	X	X	X	X	X	X
WGD-Sarsa1(λ), $\lambda=0.1$	160.7	138.4	107.1	115.9	114.9	115.1	135.2	110.4	128.9
WGD-Sarsa1(λ), $\lambda=0.6$	163.2	125.0	110.6	112.0	125.2	113.0	148.2	X	X
WGD-Sarsa2(λ), $\lambda=0.1$	173.3	158.2	112.7	133.1	124.9	110.9	115.6	115.9	122.1
WGD-Sarsa2(λ), $\lambda=0.6$	137.2	139.7	119.8	126.4	114.4	107.9	128.7	106.7	X
AC, $\alpha^\theta=0.01$	174.0	X	X	X	X	X	X	X	X
WGD-AC, $\alpha^\theta=0.01$	170.7	135.5	133.1	128.0	125.1	127.8	129.9	130.0	130.4

表2 Cartpole 实验中不同步长参数取值下各算法的平均收敛效果数据统计
Table 2 Statistics on the average convergence performance of various algorithms under asynchronous long parameter values in the Cartpole experiment

算法	$(\alpha \backslash \alpha^m)$								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sarsa	793.5	124.1	X	X	X	X	X	X	X
WGD-Sarsa	372.7	585.7	744.5	681.0	747.2	804.4	743.6	802.0	686.6
Sarsa(λ), $\lambda=0.1$	810.6	156.0	X	X	X	X	X	X	X
Sarsa(λ), $\lambda=0.9$	57.51	24.59	X	X	X	X	X	X	X
WGD-Sarsa1(λ), $\lambda=0.1$	411.2	600.0	730.6	734.3	790.6	722.5	812.7	844.9	590.4
WGD-Sarsa1(λ), $\lambda=0.9$	776.9	796.2	773.6	629.1	88.96	13.56	10.60	12.38	X
WGD-Sarsa2(λ), $\lambda=0.1$	418.2	614.9	777.3	654.7	744.5	893.9	717.2	769.7	773.1
WGD-Sarsa2(λ), $\lambda=0.9$	888.7	978.8	992.4	989.8	986.6	940.5	368.9	32.58	25.06
AC, $\alpha^0=0.1$	739.2	23.82	21.78	X	X	X	X	X	X
AC, $\alpha^0=0.9$	744.6	21.60	21.98	X	X	X	X	X	X
WGD-AC, $\alpha^0=0.1$	241.2	621.2	673.9	582.4	709.9	752.2	777.8	747.3	711.7
WGD-AC, $\alpha^0=0.9$	220.4	618.4	680.3	562.3	709.8	766.4	826.7	751.2	704.1

Cartpole 实验中不同步长参数取值下各算法的平均收敛效果数据统计, 表中数据可用于分析步长参数在不同取值的情况下对算法收敛性的影响。其中, 数字为算法所得的平均步数, X 表示算法处于发散的状态。为了方便实验的对比, 表中并未显示出全部的数据。但从表中数据可知, 虽然提高 α 的参数值可以加快 Sarsa 算法的收敛速度, 但是算法的收敛值也会变差, 甚至导致算法发散。而 WGD 方法的引入不仅有效地降低了算法发散的可能性, 也提高了算法的学习性能。Sarsa(λ) 算法中的参数 α 取到 0.3 时, 算法会发散。在 Mountain Car 实验中, WGD-Sarsa1(λ) 算法和 WGD-Sarsa2(λ) 算法的参数 α 取到 0.8 和 0.9 时, 才会发散。在 Cartpole 实验中, WGD-Sarsa1(λ) 算法中的参数 α 取到 0.9 时才会发散, 而 WGD-Sarsa2(λ) 算法仅仅只是收敛效果变差, 并没有出现发散的情况。因此引入 WGD 方法的 Sarsa(λ) 算法拥有更为广泛的取值范围, 极大地避免了算法的发散。AC 算法对参数的取值也极为严格, 取值不当会降低算法的收敛效果, 甚至导致算法发散。然而, 使用 WGD 改进后的 AC 算法能有效地提升评论家的评估能力, 不仅降低了算法发散的可能性, 还提高了算法的学习性能。

通过上述实验可以得出, WGD 方法不仅可

以有效地应用于基于梯度下降的各种算法, 并且可以对算法进一步地优化。在实际情况下, 对状态的特征进行提取时极难保证特征向量之间的线性无关性。虽然 WGD 方法在理论上对特征向量的要求较高, 但是在实验中, 算法依旧取得了令人满意的结果, 而且优化后的新算法不仅拥有更广泛的取参范围, 降低算法发散的可能性, 而且拥有更快的收敛速度和更好的收敛效果。

4 结束语

本文针对梯度下降方法的不足, 提出了权重梯度下降方法。该方法利用最小二乘方法求解期望权重, 并结合了 TD 的思想, 将时序差分评估值误差转换为权重误差, 最后利用梯度下降的方式更新权重。WGD 方法可以有效地对其他基于梯度下降的函数逼近算法进行改进, 在降低算法发散的可能性的同时, 保证算法可以收敛到正确的值上。实验结果表明, 利用 WGD 方法改后的算法, 不仅拥有更广泛的参数取值空间, 而且拥有良好的收敛效果和较快的收敛速度。

WGD 方法拥有良好的拓展性, 理论上各种针对梯度下降方法的优化手段同样可以与 WGD 方法进行结合, 从而形成新的算法, 如环境模型设置、自适应步长参数设置、冲量梯度方法等,

可以将这些改进工作作为未来的考虑对象。同时,继续对 WGD 方法进行优化,将其应用于策略梯度方法、其他非线性函数逼近方法以及各种神经网络的反向传播方法。

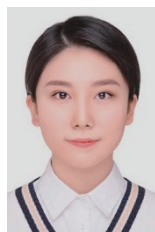
参考文献:

- [1] SUTTON R S, BARTO A G. Reinforcement learning: an introduction[J]. IEEE Transactions on Neural Networks, 1998, 9(5): 1054-1054.
- [2] 陈兴国, 俞扬. 强化学习及其在电脑围棋中的应用[J]. 自动化学报, 2016, 42(5): 685-695.
CHEN X G, YU Y. Reinforcement learning and its application to the game of GO[J]. Acta Automatica Sinica, 2016, 42(5): 685-695.
- [3] HU Y J, GAO Y, AN B. Multiagent reinforcement learning with unshared value functions[J]. IEEE Transactions on Cybernetics, 2015, 45(4):647-662.
- [4] POLYDOROS A S, NALPANTIDIS L. Survey of model-based reinforcement learning: applications on robotics[J]. Journal of Intelligent & Robotic Systems, 2017, 86(2):1-21.
- [5] SENGONZI C, KOGEDA O P, OLWAL T O. A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization[R]. 2022.
- [6] WU Y, MOZIFIAN M, SHKURTI F. Shaping rewards for reinforcement learning with imperfect demonstrations using generative models[C]//2021 IEEE International Conference on Robotics and Automation (ICRA). 2021: 6628-6634.
- [7] XU XIN, ZUO LEI, HUANG Z H. Reinforcement learning algorithms with function approximation: Recent advances and applications[J]. Information Sciences, 2014, 261(5):1-31.
- [8] GRONDMAN I, BUSONI L, LOPES G A D, et al. A survey of actor-critic reinforcement learning: standard and natural policy gradients[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6):1291-1307.
- [9] VAN-SEIJEN H, MAHMOOD A R, PILARSKI P M, et al. True online temporal-difference learning[J]. Journal of Machine Learning Research, 2015, 17(1):5057-5096.
- [10] LI K, BURDICK J W. A function approximation method for model-based high-dimensional inverse reinforcement learning[R]. 2017.
- [11] FUJIMOTO S, GU S S. A minimalist approach to offline reinforcement learning[C]//Advances in Neural Information Processing Systems, 2021, 34: 20132-20145.
- [12] THOMAS P S, BRUNSKILL E. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines[R]. 2017.
- [13] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518 (7540): 529-533.
- [14] 郭潇逍, 李程, 梅俏竹. 深度学习在游戏中的应用[J]. 自动化学报, 2016, 42(5): 676-684.
GUO X X, LI C, MEI Q Z. Deep learning applied to games[J]. Acta Automatica Sinica, 2016, 42(5): 676-684.
- [15] GEIST M, PIETQUIN O. Parametric value function approximation: a unified view[C]//Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning. 2011.
- [16] FAIRBANK M, ALONSO E. The divergence of reinforcement learning algorithms with value-iteration and function approximation[C]// International Joint Conference on Neural Networks. 2012.
- [17] BHANDARI J, RUSSO D, SINGAL R. A finite time analysis of temporal difference learning with linear function approximation[R]. 2018.
- [18] AWHEDA M D, SCHWARTZ H M. A residual gradient fuzzy reinforcement learning algorithm for differential games[J]. International Journal of Fuzzy Systems, 2017.
- [19] BO L, JI L, GHAVAMZADEH M, et al. Proximal gradient temporal difference learning algorithms[C]//International Joint Conference on Artificial Intelligence. 2016.

[作者简介]



秦晓燕(1984—),女,江苏泰州人,苏州高博软件技术职业学院副教授,主要研究方向为软件工程、人工智能。



刘禹含(2000—),女,黑龙江大庆人,加拿大滑铁卢大学硕士生,主要研究方向为数字媒体技术。



徐云龙(1964—),男,江苏苏州人,苏州大学副教授,主要研究方向为机器学习、操作系统。



李斌(1994—),男,江苏镇江人,主要研究方向为强化学习。