doi:10.3969/j.issn.1000-436x.2013.01.009

基于自适应势函数塑造奖赏机制的梯度下降 Sarsa(?)算法

肖飞[1]，刘全[1,2]，傅启明[1]，孙洪坤[1]，高龙[1]
（1．苏州大学 计算机科学与技术学院，江苏 苏州 215006；2．吉林大学 符号计算与知识工程教育部重点实验室，吉林 长春 130012）

摘 要：针对连续状态空间下的强化学习算法初始性能差及收敛速度慢的问题，提出利用自适应势函数塑造奖赏机制来改进强化学习算法。该机制通过额外的奖赏信号自适应地将模型知识传递给学习器，可以有效提高算法的初始性能及收敛速度。鉴于径向基函数(RBF)网络的优良性能及存在的问题，提出利用自适应归一化 RBF(ANRBF)网络作为势函数来塑造奖赏。基于 ANRBF 网络提出了梯度下降(GD)版的强化学习算法——ANRBF-GD-Sarsa(?)。从理论上分析了 ANRBF-GD-Sarsa(?)算法的收敛性，并通过实验验证了 ANRBF-GD-Sarsa(?)算法具有较好的初始性能及收敛速度。

关键词：强化学习；Sarsa(?)；梯度下降；势函数；塑造奖赏
中图分类号：TP181 **文献标识码：**A **文章编号：**1000-436X(2013)01-0077-12

Gradient descent Sarsa(?) algorithm based on the adaptive potential function shaping reward mechanism

XIAO Fei[1], LIU Quan[1, 2], FU Qi-ming[1], SUN Hong-kun[1], GAO Long[1]
(1. Institute of Computer Science and Technology, Soochow University, Suzhou 215006, China;
2. Key Laboratory of Symbolic Computation and Knowledge E gineering of Ministry of Education, Jilin University, Changchun 130012, China)

Abstract: In the reinforcement leaning tasks with continuous state spaces, the algorithms are usually facing the problems of ill initial performance and low convergence speed. In order to solve these problems, the potential function shaping reward mechanism was proposed to improve the reinforcement learning algorithms. This mechanism propagates model knowledge to the learner adaptively in the form of the additional reward signal, so that the initial performance and convergence speed could be improved effectively. In view of the good performance and existing problems of the radial basis function (RBF) network, the adaptive normalized RBF (ANRBF) network was put forward to use as a potential function to generate the shaping rewards. A gradient descent (GD) algorithm named ANRBF-GD-Sarsa(?) was proposed based on the ANRBF network. The convergence of ANRBF-GD-Sarsa(?) algorithm was analyzed theoretically. Extensive experiments are conducted to show the good initial performance and high convergence speed of the proposed algorithm.

Key words: reinforcement learning; Sarsa(?); gradient descent; potential function; shaping reward

1 引言

强化学习中 Agent 通过试错(trail-and-error)与环境进行交互。通过时间信度分配(temporal credit assignment)机制将每个时间步获得的延迟回报(delayed reward)传递给过去动作序列中的某些动作。用值函数评价每个状态或状态动作对的好坏程度，最终通过值函数确定到达目标的最优策略。强

收稿日期：2012-08-23；**修回日期：**2012-11-20
基金项目：国家自然科学基金资助项目(61070223, 61103045, 61070122, 61272005)；江苏省自然科学基金资助项目(BK2012616)；江苏省高校自然科学研究基金资助项目(09KJA520002, 09KJB520012)；吉林大学符号计算与知识工程教育部重点实验室基金资助项目(93K172012K04)
Foundation Items: The National Natural Science Foundation of China(61070223, 61103045, 61070122, 61272005); The Natural Science Foundation of Jiangsu Province(BK2012616); The High School Natural Foundation of Jiangsu Province(09KJA520002, 09KJB520012); The Foundation of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University(93K172012K04)

化学习的自学习和在线学习的特性决定了其非常适用于解决复杂的、环境模型未知的、不确定的时序决策优化问题[1]。随着强化学习理论研究的不断深入，强化学习方法被越来越多地用于制造过程优化、机器人控制、游戏等领域[1~3]。

传统的强化学习方法利用二维表存储值函数，并不断进行修改，直至值函数收敛。每一个表项是状态或状态动作对及对应的估计值。这种方法清晰简单，非常适用于离散的小状态空间问题。但当处理大状态或连续状态空间问题时，会面临收敛速度慢甚至无法收敛，即“维数灾”问题[1]。解决强化学习“维数灾”问题主要有3种途径。1) 通过聚类等方法对状态空间进行离散化并降维，再利用传统的表格式强化学习方法解决问题[4]。然而在离散化操作中，如果离散化的精度太高，那么状态数会以指数级的速度增长，增加了时间和空间复杂度。如果离散化的精度太低，又无法保证所估计值函数的精度。因此该方法面临离散化精度和状态空间复杂度的权衡问题，而且离散化后无法保证收敛到原问题的全局最优解。2) 基于任务分解的思想，采用分层和并行等技术来改进强化学习方法，充分利用抽象化思想和计算资源来解决强化学习问题[5,6]。如何进行任务分解及并行方法中多个Agent间如何协作等是该类方法的难点。3) 将强化学习与函数逼近相结合，利用一个近似函数来对强化学习中的值函数进行建模，通过样本逼近真实的值函数模型[7,8]。在结合函数逼近的强化学习方法中，学习的经验信息能够从状态空间的子集泛化至整个状态空间。Agent利用近似函数选择每个状态下的最优动作。

早在20世纪90年代，Tsitsiklis及Sutton等人就对结合函数逼近的强化学习方法进行了探索和研究[9,10]。近年来，利用函数逼近来解决强化学习的“维数灾”问题已经逐渐成为了研究的热点。2009年，Bonarini等人基于模糊逻辑提出了一种模糊Q学习算法，并在模糊推理系统中验证了算法的有效性[11]。Heinen等人于2010年提出利用增量式概率神经网络来逼近强化学习问题的值函数，通过实验验证了算法的性能[12]。Maei等人于2010年提出了结合资格迹(eligibility trace)的GQ(*?*)(general Q(*?*))算法，能够很好地实现时间差分(TD, temporal difference)预测学习[13]。同年，Maei等人将GQ(*?*)算法中的目标策略设为贪心策略，*?*设为0，提出了离策略Greedy-GQ算法，用于解决强化学习的控制问题，并通过Baird反例验证了算法的有效性[14]。函数逼近可以有效提高学习算法的泛化能力，对连续状态空间问题适应性很强。然而，强化学习方法通过反向传播延迟回报机制来解决时间信度分配问题，TD学习方法中将每一步获得的奖赏反向传递给过去动作序列中的某些动作，不断进行迭代。此学习方法效率较低，在学习初期，需要较多的探索后才能获取第一个有效奖赏。扩展到大状态空间或者连续状态空间时，此问题会更加突出。针对该问题，一种被证明行之有效的方法是利用塑造奖赏机制，将模型知识以奖赏的形式反馈给学习器，减少次优动作的选择次数，从而加快学习系统的收敛速度。Randløv等人指出了如果塑造奖赏机制使用不当，将会对学习过程产生误导[15]。Ng等人研究了融入塑造奖赏机制的算法保持策略不变性的充要条件[16]。如何构造塑造奖赏以及如何及时准确地将模型知识传递给学习器，是这类方法的关键之处。径向基函数(RBF, radial basis function)网络利用一组RBF将输入空间的非线性逼近问题转化为特征空间的线性逼近问题[17]。RBF网络不仅可以有效解决输入空间的非线性逼近问题，又保持了线性逼近模型的简单性及强泛化能力。在处理具有连续状态空间的强化学习问题时，RBF网络非常适合用来构造塑造奖赏机制中的势函数。

强化学习的在线特性决定了RBF网络逼近模型会面临“灾难性扰动”问题，即新样本作用于学习模型后非常容易对之前学习到的输入输出映射关系产生破坏。本文提出利用自适应归一化径向基函数(ANRBF, adaptive normalized radial basis function)网络作为势函数来塑造奖赏。在RBF网络中融入归一化和自适应机制后，能够提供一个更加平滑且可微的势函数。ANRBF网络自适应地将模型知识传递给学习器，可以有效提高算法的初始性能及收敛速度。线性梯度下降方法简单、泛化能力强，适用于连续状态空间强化学习问题，而且在满足一定条件下收敛性可以得到保证。基于ANRBF网络提出了梯度下降版的强化学习算法——ANRBF-GD-Sarsa(*?*)。本文从理论上分析了ANRBF-GD-Sarsa(*?*)算法的收敛性。将ANRBF-GD-Sarsa(*?*)算法应用于Mountain Car仿真平台，实验结果表明，ANRBF-GD-Sarsa(*?*)算法具有较好的初始性能及收敛速度。

2　相关理论

2.1　Sarsa(?)算法

传统的强化学习方法考虑的是具有离散状态空间和离散动作空间的强化学习问题[1]。当问题的状态空间扩展到连续空间时，原有的一些概念的定义形式将不再适用，本文需要对这些概念进行重新定义。具体定义如下。

定义 1　状态向量。状态是强化学习系统中的重要元素，是表征系统某一时刻特征及属性的最少数目变量的有序集合，以向量形式表示。假设系统状态由 n 个实数变量描述，则在 t 时刻系统的状态向量表示为 $\boldsymbol{x}_t=[x_1(t),x_2(t),\L,x_n(t)]^{\mathrm{T}}\in ¡^{n}$。在不关心时刻的情况下，统一用 $\boldsymbol{x}=[x_1,x_2,\L,x_n]^{\mathrm{T}}$ 来表示状态向量。

定义 2　状态空间。强化学习问题 P 的状态空间为该问题所有可能状态的集合。假设系统状态表示为 n 维状态向量 $\boldsymbol{x}=[x_1,x_2,\L,x_n]^{\mathrm{T}}$，则 P 的状态空间即为以状态向量各分量为坐标轴而形成的 n 维向量空间的某个子空间，记作 $X(P)\subseteq ¡^{n}$。形式化定义为 $X(P)=\{\boldsymbol{x}\mid \boldsymbol{x}=[x_1,x_2,\L,x_n]^{\mathrm{T}},x_i\in D_i,i=1,\L,n\}$ 其中，D_i 是状态向量第 i 个分量的值域。在不强调特定问题时，用 X 表示状态空间。$\forall D_i$，若至少一个为连续域，那么 X 称为连续状态空间；若全部为连续域，X 称为完全连续状态空间；若无连续域，X 称为离散状态空间。

仿照定义 1 和定义 2，可以定义动作向量和动作空间。本文关注的是具有连续状态空间和离散动作空间的强化学习问题。若无特别说明，本文统一用 $U=\{u_i\}_{i=1}^{k},k\geq 2$ 来表示问题的动作空间。

用强化学习方法解决实际问题，主要分为预测问题和控制问题 2 部分。TD 学习方法作为强化学习中的一类很重要的算法，在各种学习任务中都能很好地解决预测和控制问题。强化学习算法收敛的必要条件是所有状态或状态动作对被无限次访问，即需要平衡探索与利用。可以通过在策略(on-policy)和离策略(off-policy) 2 类方法来平衡探索和利用[1]。Sarsa 算法是经典的在策略 TD 控制学习算法，传统的 Q 学习和 Sarsa 算法只向前看一步或者 TD 误差仅向后传递一步。为了获得一个更高效的学习算法，需要综合考虑整个行动轨迹，于是引入资格迹，产生了更一般的 Sarsa(?)算法[1]。同样以马尔科夫决策过程为基础，通过试错来不断地与环境进行交互，最终学习到一个最优策略。在实际问题中，Sarsa(?)算法通过迭代来学习，迭代方法如式(1)

$$Q_{t+1}(\boldsymbol{x}_t,u_t)=Q_t(\boldsymbol{x}_t,u_t)+a\,d_t e_t(\boldsymbol{x}_t,u_t) \tag{1}$$

其中，$Q_t(\boldsymbol{x}_t,u_t)$ 和 $Q_{t+1}(\boldsymbol{x}_t,u_t)$ 分别为在 t 和 $t+1$ 时刻对状态动作对 $<x_t,u_t>$ 值函数 $Q^{\mathrm{p}}(\boldsymbol{x}_t,u_t)$ 的估计值。$a\in[0,1]$ 为学习率，用来控制学习速度。a 越大，学习速度越快，但越容易引起振荡。TD 误差计算方式为 $d_t=r_{t+1}+gQ_t(\boldsymbol{x}_{t+1},u_{t+1})-Q_t(\boldsymbol{x}_t,u_t)$，$r_{t+1}$ 为立即奖赏。$g\in[0,1]$ 为折扣率，表明 Agent 的远视程度，g 越大，表明越重视未来奖赏。$e_t(\boldsymbol{x}_t,u_t)$ 为状态动作对 $<x_t,u_t>$ 在 t 时刻的资格迹，更新方式如下，$\forall \boldsymbol{x}\in X,u\in U$，若 $\boldsymbol{x}=\boldsymbol{x}_t,u=u_t$，则 $e_t(\boldsymbol{x},u)=1$（替代迹）或 $e_t(\boldsymbol{x},u)=e_t(\boldsymbol{x},u)+1$（累加迹），否则 $e_t(\boldsymbol{x},u)=gl\,e_{t-1}(\boldsymbol{x},u)$，其中，$l\in[0,1]$ 为资格迹衰减因子。当 $l=0$ 时，算法退化为经典的一步 Sarsa 算法。当 $0<l<1$ 时，l 越大 TD 误差向后传播的越远，作用的状态动作轨迹就越长。当 $l=1$ 时，Sarsa(1)近似等价于带 g 折扣的 Monte Carlo 方法，但是 Sarsa(1)可以在线和增量式实现，比带 g 折扣的 Monte Carlo 方法应用范围更广。

2.2　梯度下降方法

利用梯度下降方法来解决强化学习问题时，首先要对算法中的值函数进行线性建模[1]。模型如式(2)

$$Q_t(\boldsymbol{x},u)=?_t^{\mathrm{T}}j\ (\boldsymbol{x},u)=\sum_{i=1}^{n}q_i j_i(\boldsymbol{x},u) \tag{2}$$

其中，$?_t$ 为在 t 时刻的 n 维权值向量，$j\ (\boldsymbol{x},u)$ 为状态动作对 $<\boldsymbol{x},u>$ 的 n 维特征向量。在式(2)所示模型的基础上通过最小化均方误差(MSE, mean-squared error)来逼近最优值函数[1]。MSE 形式化定义如式(3)

$$MSE(?_t)=\sum_{\boldsymbol{x}\in X,u\in U}\hat{P}(\boldsymbol{x},u)\left[Q^{\mathrm{p}}(\boldsymbol{x},u)-Q_t(\boldsymbol{x},u)\right]^2 \tag{3}$$

其中，$\hat{P}(\cdot,\cdot)$ 为状态动作对的权值分布，用来权衡各状态动作对的真实值与估计值的误差的平方。采用 TD 方法时，$d=Q^{\mathrm{p}}(\boldsymbol{x},u)-Q_t(\boldsymbol{x},u)$ 为 TD 误差，采用梯度下降递归迭代策略来最小化 MSE[1]。过程如式(4)

$$\begin{aligned}?_{t+1}&=?_t-\frac{1}{2}a\nabla_{?_t}\left[Q^{\mathrm{p}}(\boldsymbol{x}_t,u_t)-Q_t(\boldsymbol{x}_t,u_t)\right]^2\\&=?_t+a\left[Q^{\mathrm{p}}(\boldsymbol{x}_t,u_t)-Q_t(\boldsymbol{x}_t,u_t)\right]\nabla_{?_t}Q_t(\boldsymbol{x}_t,u_t)\end{aligned} \tag{4}$$

在式(5)的条件下，若学习率a 随时间衰减，那么可以证明采用梯度下降方法的线性学习模型是收敛的[9]。

$$\sum_{t=0}^{\infty} a_t = \infty,\ \sum_{t=0}^{\infty} a_t^2 < \infty \tag{5}$$

3 自适应势函数塑造奖赏机制

定义 3 势函数(PF, potential function)。PF 来源于物理学中的势，在保守场中，单位质点在 A 点与参考点的势能之差是一定的，把这个势能差定义为保守场中 A 点的“势”。强化学习中通过一个实值函数$F: X \to$ 来表征 Agent 处于某一情境所具有的势，其中，X 为任意状态空间。$F(x)$ 越大说明状态 x 越接近目标状态，这里的实值函数F 称作势函数，具体问题需要具体定义。

定义 4 基于势函数的塑造奖赏(PF-SR, potential function based shaping reward)。基于势函数$F: X \to$ 定义一个有界实值映射 $F: X \times U \times X \to$，其中，$X$ 为任意状态空间，U 为离散动作空间，塑造奖赏定义为式(6)

$$F(\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1}) = gF(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t) \tag{6}$$

其中，$g \in [0,1]$为折扣率。塑造奖赏就是当前状态和后继状态势的折扣差值。

定义 5 基于塑造奖赏的连续状态马尔科夫决策过程(SR-CS-MDP, shaping reward based continuous state markov decision process)。SR-CS-MDP 为一个四元组$M' = \{X, U, R', T\}$，其中，X 为任意连续状态空间，U 为任意离散动作空间，$R': X \times U \times X \to$ 为在原始奖赏函数 R 的基础上融入塑造奖赏之后的奖赏函数，$R'(\boldsymbol{x}, u, \boldsymbol{x}') = R(\boldsymbol{x}, u, \boldsymbol{x}') + F(\boldsymbol{x}, u, \boldsymbol{x}')$，$T: X \times U \to PD(X)$ 为状态转移函数。在 SR-CS-MDP 中的一个典型的交互过程为：在时间步 t，Agent 感知到当前的状态 $\boldsymbol{x}_t \in X$，根据当前行为策略$p(\boldsymbol{x}_t, u_t) = p(u_t \mid \boldsymbol{x}_t)$，选择一个动作$u_t \in U$，将$u_t$ 作用于环境，环境对此做出响应，根据状态转移函数，$\boldsymbol{x}_t$ 以概率$T(\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1})$ 转移至 $\boldsymbol{x}_{t+1} \in X$，并给出立即奖赏值$R'(\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1})$。

下面以 n 维连续状态空间为例，阐述一种利用 ANRBF 网络构建势函数的方法，引入 ANRBF 网络之前需要给出状态空间高斯划分的定义，ANRBF 网络与状态空间高斯划分一一对应。

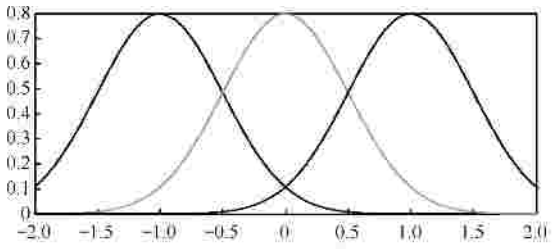
定义 6 状态空间高斯划分。连续状态空间 $X = \{\boldsymbol{x} \mid \boldsymbol{x} = [x_1, \cdots, x_n]^{\mathrm{T}}, x_i \in D_i, i = 1, \cdots, n\}$，连续域 D_i 被离散化为 m_i 个子域，$i = 1, \cdots, n$，记 $m = \prod_{i=1}^{n} m_i$，则 X 被划分为 m 个连续子空间，记为 $B_1, B_2, \cdots, B_m$。用 n 维向量 $\boldsymbol{c}_k = [c_{k1}, \cdots, c_{kn}]^{\mathrm{T}}$、$\boldsymbol{s}_k = [s_{k1}, \cdots, s_{kn}]^{\mathrm{T}}$ 分别表示第 k 个子空间的中心及宽度，$k = 1, \cdots, m$。用 m 个 n 维高斯函数$f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})$ 分别与划分所得的 m 个连续子空间 $B_1, B_2, \cdots, B_m$ 一一对应，其中，$f_k(\boldsymbol{x}) = \exp\left(-\sum_{j=1}^{n} \frac{(x_j - c_{kj})^2}{2s_{kj}^2}\right)$，$k = 1, \cdots, m$，若：

1) $B_i \neq B_j, i \neq j$，$i, j = 1, 2, \cdots, m$，即所有划分分块两两不重合；

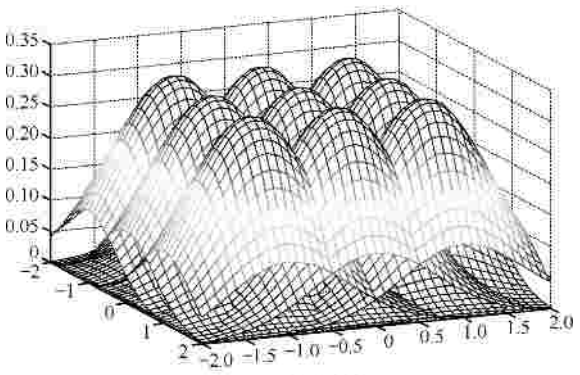
2) $W(f_i(\boldsymbol{x})) \supset B_i$，$i = 1, 2, \cdots, m$，即任意高斯函数的状态空间投影真包含对应的划分分块，其中，W 为投影算子；

3) $\bigcup_{i=1}^{m} B_i \supseteq X$，即所有划分分块的并集包含原状态空间，则称 $B_1, B_2, \cdots, B_m$ 为状态空间的一个高斯划分，$f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})$ 为各分块的基函数。

下面通过 2 个例子来说明状态空间高斯划分，如图 1 所示，图 1(a)和图 1(b)分别为利用 3 个一维和 9 个二维等距高斯函数平均划分状态空间 $X = \{\boldsymbol{x} \mid \boldsymbol{x} = [x_1], x_1 \in [-2, 2]\}$ 和 $X = \{\boldsymbol{x} \mid \boldsymbol{x} = [x_1, x_2]^{\mathrm{T}}, x_1, x_2 \in [-2, 2]\}$ 的划分示意图。



(a) 一维状态空间



(b) 二维状态空间

图 1　状态空间高斯划分

基于上述状态空间高斯划分，可以导出对应的 RBF 网络。在此基础上，将状态空间高斯划分确定的基函数进行归一化处理，然后引入自适应机制，构建一个 ANRBF 网络。具体定义如下。

定义 7　ANRBF 网络。ANRBF 网络是一种 $n\times m\times 1$ 的三层归一化带反馈机制的神经网络，它可在线自适应地调整隐藏层各节点的权值。ANRBF 网络的结构如图 2 所示。

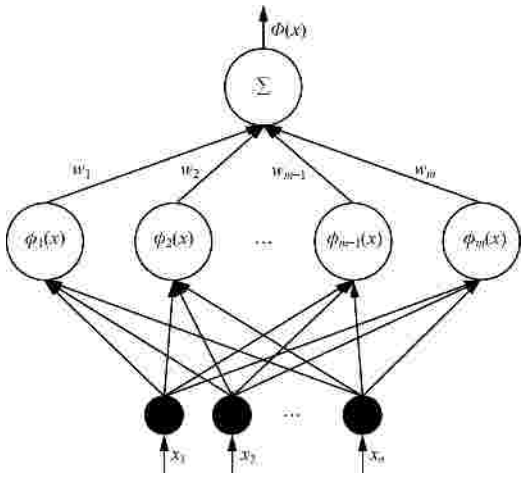


图 2　ANRBF 网络结构

1) 最底层称为输入层，表示 n 维输入向量，这里是状态向量 $\boldsymbol{x}=[x_1,x_2,\L,x_n]^{\mathrm{T}}\in j^{n}$。

2) 中间层称为隐藏层，共 m 个隐节点。每个隐节点内置一个激活函数，这里采用高斯 RBF。隐节点激活函数定义如式(7)

$$f_i(\boldsymbol{x})=j_i(\boldsymbol{x})\left[\sum_{k=1}^{m}j_k(\boldsymbol{x})\right]^{-1}$$

$$j_i(\boldsymbol{x})=\exp\left(-\sum_{j=1}^{n}\frac{(x_j-c_{ij})^2}{2s_{ij}^2}\right),\ i=1,2,\L,m \quad (7)$$

这里的激活函数就是对应高斯划分中的基函数经过归一化而得到的，其中，n 维向量 $\boldsymbol{c}_i=[c_{i1},\L,c_{in}]^{\mathrm{T}}$、$\boldsymbol{s}_i=[s_{i1},\L,s_{in}]^{\mathrm{T}}$ 分别表示状态空间高斯划分中的第 i 个子空间的中心向量及其宽度。

3) 最上层称为输出层，对隐藏层各节点进行加权求和，表示为 $F(\boldsymbol{x})=\sum_{i=1}^{m}w_i f_i(\boldsymbol{x})$，其中，$w_i$ 是经状态空间高斯划分后生成的各个子空间分块的权重。隐藏层的每个激活函数和输出权值，与对应的状态空间划分紧密耦合，知识就是通过这样的一种耦合机制进行传递的。

4) 调整输出层权值 $\boldsymbol{w}=[w_1,w_2,\L,w_m]^{\mathrm{T}}$。对隐藏层各激活函数的输出进行线性加权求和，得到势函数，形式化描述为 $F(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}f(\boldsymbol{x})$。利用梯度下降方法来调整权值 $\boldsymbol{w}$，具体方法如下

$$\boldsymbol{w}=\boldsymbol{w}+b\,d\nabla_{\boldsymbol{w}}F(\boldsymbol{x})=\boldsymbol{w}+b\,df(\boldsymbol{x})$$

其中，d 为学习过程中产生的 TD 误差；$b\in[0,1]$ 为步长参数，控制梯度调整的速度，b 越大，调整越快，但越容易引起震荡；梯度向量为

$$\nabla_{\boldsymbol{w}}F(\boldsymbol{x})=\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{w}}=\frac{\partial(\boldsymbol{w}^{\mathrm{T}}f(\boldsymbol{x}))}{\partial \boldsymbol{w}}=f(\boldsymbol{x})$$

其中，$f(\boldsymbol{x})=[f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\L,f_m(\boldsymbol{x})]^{\mathrm{T}}$ 为输入向量 $\boldsymbol{x}$ 经过隐藏层激活函数编码后得到的状态特征向量。

4　自适应势函数塑造奖赏机制下的梯度下降 Sarsa(?)算法

4.1　算法框架

本文所提出的算法可构建为一个行动者-评论家(actor-critic)模型，算法框架如图 3 所示。行动者(actor)感知到环境(environment)的当前状态(state)，根据不断修正优化的策略(policy)来选择一个动作(action)，并将此动作作用于环境，环境对此做出响应，给出一个立即奖赏(reward)，环境状态转移至一个后继状态。评论家模型包含 2 个评论子模型，一个是利用梯度下降的值函数(value function)模型，另一个是产生塑造奖赏的势函数(potential function)模型。值函数模型在收集到状态、动作、奖赏等信息后，计算出 TD 误差 d_{old}。势函数模型在收集到状态、动作、奖赏等信息后，计算出塑造奖赏 SR。然后利用 SR 重塑(remodel)TD 误差，得到新的 TD 误差 d_{new}。d_{new} 反作用于 2 个评论子模型以及策略模块，评论子模型及策略模块根据 d_{new} 进行修正优化。算法按上述流程不断循环，直至收敛到一个最优策略。

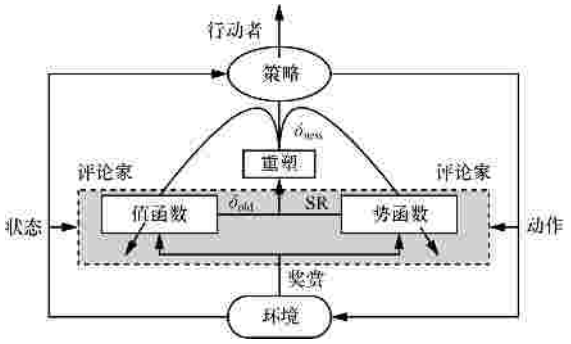


图 3 算法框架

4.2 ANRBF-GD-Sarsa(?)算法

考虑具有连续状态空间 X 和离散动作空间 U 的强化学习问题。值函数模型为 $Q(x,u)=\ ?^{\mathrm{T}}j\ (x,u)$，其中，$?\in ¡^{h}$，$j\ (\cdot,\cdot)\in ¡^{h}$。势函数模型为 $F\ (x)=w^{\mathrm{T}}f\ (x)$，其中，$w\in ¡^{m}$，$f\ (\cdot)\in ¡^{m}$。算法详细描述如下。

算法 1 ANRBF-GD-Sarsa(?)：基于 ANRBF 网络塑造奖赏机制的梯度下降 Sarsa(?)算法

1) 状态空间进行高斯划分，初始化 ANRBF 网络隐藏层各节点的激活函数 $f\ (\cdot)=[f_1(\cdot),f_2(\cdot),\mathrm{L},f_m(\cdot)]^{\mathrm{T}}$；

2) 构建 ANRBF 网络势函数模型 $F\ (\cdot)=w^{\mathrm{T}}f\ (\cdot)$，初始化权值向量 $w=\vec{0}\in ¡^{m}$；

3) 构建带有 Hash 机制的 10 个 8×8 的 Tiling 编码模块；

4) 构建线性值函数模型 $Q(\cdot,\cdot)=?^{\mathrm{T}}j\ (\cdot,\cdot)$，初始化权值及资格迹向量 $?=\vec{0}\in ¡^{h}$，$e=\vec{0}\in ¡^{h}$；

5) 重复（对每一个情节）；

6) 初始化当前状态 x 及动作 u；

7) 重复（对该情节中的每一步）；

8) 执行动作 u，观察 r，x'；

9) 将数据 $<x',U(x')>$ 输入当前值函数模型；

10) $u'\leftarrow$ 通过动作选择策略选择状态 x' 下的动作；

11) 收集数据 $<x,u,x',r,u'>$；

12) 计算 TD 误差，$d_{\mathrm{old}}=r+gQ(x',u')-Q(x,u)$；

13) 计算 SR，$F(x,u,x')=g\Phi(x')-F\ (x)$；

14) 计算新 TD 误差，$d_{\mathrm{new}}=d_{\mathrm{old}}+F\ (x,u,x')$；

15) 更新资格迹，$e\leftarrow gl\ e+\nabla_w Q(x,u)=gl\ e+j\ (x,u)$；

16) 将 d_{new} 反馈给值函数模型及势函数模型；

17) 调整权值，$?\leftarrow ?+ad_{\mathrm{new}}e$，$w\leftarrow w+bd_{\mathrm{new}}f(x)$；

18) $x=x'$，$u=u'$；

19) 直到 x 是终止状态；

20) 直到运行完设定情节数或满足其他终止条件。

4.3 Tile 编码机制

上一节描述的 ANRBF-GD-Sarsa(?)算法中的 $j\ (x,u)$ 为状态动作对 $<x,u>$ 的特征向量。本算法的值函数模型首先利用 Tile 编码机制来对状态 x 进行编码，得到状态 x 的特征向量。然后根据不同的动作，利用 Hash 机制将得到的状态特征向量映射到 h 维的状态动作对的特征向量空间，从而得到 $<x,u>$ 的特征向量 $j\ (x,u)$。

Tile 编码简单高效，能够高度契合在线学习任务的特性[18]。下面详细描述利用 Tile 编码机制提取状态特征向量的过程：假设状态空间为 n 维矩形区域，按照随机偏移规则，采用 $i\ (\geq 1)$ 个比状态空间略大的 n 维矩形区域重叠覆盖整个状态空间。这里的矩形区域称为 Tiling，共有 i 个 Tiling。每个 Tiling 被划分为 $j\ (\geq 1)$ 个 Tile，每个 Tile 表示的是状态特征向量中的某一特征分量的接受区域。这里得到的状态特征向量为 $i\times j$ 维。如果状态 x 出现在第 $k\ (1\leq k\leq i\times j)$ 个 Tile 的接受区域中，则将 x 的特征向量中的第 k 个分量置为 1。编码完成后，$i\times j$ 维的状态特征向量中仅有 i 个分量被置为 1，其余都为 0。

以任意 2 维连续状态空间为例，Tile 编码模型如图 4 所示。图中横坐标表示状态的第 1 维，纵坐标表示状态的第 2 维，阴影部分为状态空间，阴影部分中的每一个点都是一个状态。使用 10 个 8×8 的 Tiling 来编码，10 个 Tiling 按照随机偏移规则来覆盖状态空间。为了方便起见，图中只画了 2 个 Tiling。状态编码过程如下：在设置好的 10 个 Tiling 中，查找状态落在哪 10 个 Tile 的接受区域，在对应的 640 维二进制编码中将对应位置置 1，其余位置置 0。经过 Tiling 编码得到的状态特征向量具有一定的稀疏性，这样不仅可以有效控制特征向量的有效长度，而且有利于降低计算复杂度。Tile 编码中使用的 Tiling 数目越多，编码的精度就越高，但计算复杂度也会越高。

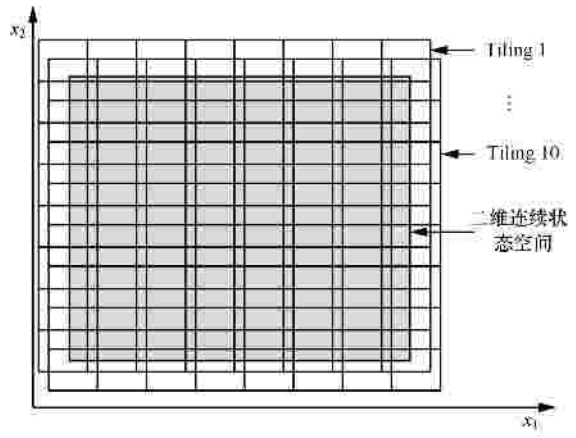


图 4　状态空间 Tile 编码

4.4　算法收敛性分析

下面从误差更新的角度切入，与经典的 GD-Sarsa(?)算法[1]进行比较，分析 ANRBF-GD-Sarsa(?)算法的收敛性，并证明 ANRBF-GD-Sarsa(?)算法与利用势函数进行初始化的 GD-Sarsa(?)算法具有一致的收敛性。

定理 1　在确定性马尔科夫决策过程中，如果算法具有相同的学习经验序列，那么 ANRBF-GD-Sarsa(?)算法与利用势函数进行初始化的 GD- Sarsa(?)算法具有一致的收敛性。

证明　分别用 L 和 $\tilde{L}$ 表示 GD-Sarsa(?)算法和 ANRBF-GD-Sarsa(?)算法的学习器。考虑具有连续状态空间 X 和离散动作空间 U 的强化学习问题。$\forall x\in X, u\in U$，L 的值函数建模为 $Q(x,u)=?^{\mathrm{T}}j(x,u)$，$\tilde{L}$ 的值函数建模为 $\tilde{Q}(x,u)=\tilde{?}^{\mathrm{T}}j(x,u)$。初始化为 $Q(x,u)=Q_0(x,u)+F(x)$，$\tilde{Q}(x,u)=Q_0(x,u)$，其中，$Q_0(x,u)=?_0^{\mathrm{T}}j(x,u)$，$F(x)$ 为势函数。假设在状态 x 下采取动作 u 转移到状态 x'，获得回报 r，在 x' 下根据行为策略选择动作 u'，那么可以得到一个经验五元组 $<x,u,x',r,u'>$。L 和 $\tilde{L}$ 分别更新各自的权值，更新如式(8)和式(9)

$$?\leftarrow ?+a\underbrace{\left[r+gQ(x',u')-Q(x,u)\right]}_{dQ(x,u)}e \tag{8}$$

$$\tilde{?}\leftarrow \tilde{?}+a\underbrace{\left[r+g\tilde{Q}(x',u')-\tilde{Q}(x,u)+F(x,u,x')\right]}_{d\tilde{Q}(x,u)}\tilde{e} \tag{9}$$

其中，$dQ(x,u)$、$d\tilde{Q}(x,u)$ 为 TD 误差；e、$\tilde{e}$ 为对应逼近模型的资格迹向量。状态动作对 $<x,u>$ 的当前值函数与初始值函数的差分别记为 $\Delta Q(x,u)$、$\Delta\tilde{Q}(x,u)$。将式(8)和式(9)权值更新映射到 Q 值函数的更新，Q 值函数变化量如式(10)和式(11)

$$\begin{aligned}\Delta Q(x,u)&=Q(x,u)-Q_0(x,u)-F(x)\\&=(a\,dQ(x,u)e)^{\mathrm{T}}j(x,u)\end{aligned} \tag{10}$$

$$\begin{aligned}\Delta\tilde{Q}(x,u)&=\tilde{Q}(x,u)-Q_0(x,u)\\&=(a\,d\tilde{Q}(x,u)\tilde{e})^{\mathrm{T}}j(x,u)\end{aligned} \tag{11}$$

假定现在 L 和 $\tilde{L}$ 具有相同的学习经验序列，利用归纳法证明 L 和 $\tilde{L}$ 等价，即要证明 $\forall x\in X$，$u\in U$，$\Delta Q(x,u)=\Delta\tilde{Q}(x,u)$。归纳证明如下。

1) 当值函数 $Q(x,u)$ 和 $\tilde{Q}(x,u)$ 都是初始模型时，即 $\Delta Q(x,u)=\Delta\tilde{Q}(x,u)=0$。

2) 假设到目前为止，$\forall x\in X, u\in U$，有 $\Delta Q(x,u)=\Delta\tilde{Q}(x,u)$。在此基础上，当前时间步获得了一个新的经验五元组 $<x,u,x',r,u'>$，此时 TD 误差计算如下

$$\begin{aligned}dQ(x,u)&=r+gQ(x',u')-Q(x,u)\\&=r+g\left(Q_0(x',u')+F(x')+\Delta Q(x',u')\right)-\\&\quad Q_0(x,u)-F(x)-\Delta Q(x,u)\\&=r+g\left(Q_0(x',u')+\Delta Q(x',u')\right)-\\&\quad Q_0(x,u)-\Delta Q(x,u)+gF(x')-F(x),\end{aligned}$$

$$\begin{aligned}d\tilde{Q}(x,u)&=r+g\tilde{Q}(x',u')-\tilde{Q}(x,u)+F(x,u,x')\\&=r+g\left(Q_0(x',u')+\Delta\tilde{Q}(x',u')\right)-Q_0(x,u)-\\&\quad\Delta\tilde{Q}(x,u)+gF(x')-F(x)\\&=r+g\left(Q_0(x',u')+\Delta Q(x',u')\right)-Q_0(x,u)-\\&\quad\Delta Q(x,u)+gF(x')-F(x)\end{aligned}$$

基于上述计算过程，可得 $d\tilde{Q}(x,u)=dQ(x,u)$，即在此新经验的作用下 L 和 $\tilde{L}$ 获得的 TD 误差相同。另一方面，由于具有相同的经验序列，L 和 $\tilde{L}$ 资格迹同步更新，又由于每个时间步 L 和 $\tilde{L}$ 获得的 TD 误差相同，因此，L 和 $\tilde{L}$ 资格迹每个时间步同量更新，所以对任一时间步有 $e=\tilde{e}$。从而根据式(10)和式(11)可得，$\forall x\in X, u\in U$，$\Delta Q(x,u)=\Delta\tilde{Q}(x,u)$。综上所述，在相同经验下，$L$ 和 $\tilde{L}$ 等价，即证明了在确定性 MDP 中，若具有相同的学习经验序列，那么 ANRBF-GD-Sarsa(?)算法与利用势函数进行初始化的 GD-Sarsa(?)算法具有一致的收敛性。

定理 2　在式(5)所示的条件下，如果学习率 a 可以随时间衰减，那么本文所提的 ANRBF-GD-Sarsa(?)算法至少能够收敛到原问题的一个局部最优解。

证明　从文献[9]得知，强化学习中采用梯度下

降训练方法的线性学习算法，比如 Sutton 等提出的 GD-Sarsa(?)算法，在满足式(5)的条件下，如果学习率 α 可以随时间衰减，那么能够保证该算法至少能够收敛到原问题的一个局部最优解。又因为定理 2 证明了在具有相同经验序列的情况下，ANRBF-GD-Sarsa(?)算法与利用势函数初始化的 GD-Sarsa(?)算法具有一致的收敛性。而强化学习算法中任意初始化的值函数都不会改变算法的收敛结果，因此，ANRBF-GD-Sarsa(?)算法在满足式(5)的条件下，如果学习率 α 可以随时间衰减，至少能够收敛到原问题的一个局部最优解。

5 实验及结果分析

5.1 实验描述

为了验证所提算法的性能，针对具有二维连续状态空间与一维离散动作空间的 Mountain Car 问题进行仿真研究。Mountain Car 问题除了系统的状态观测值以外，没有任何有关系统动力学模型的先验知识，因此难以采用传统的基于模型的最优化控制方法进行求解。图 5 给出了 Mountain Car 问题的示意。

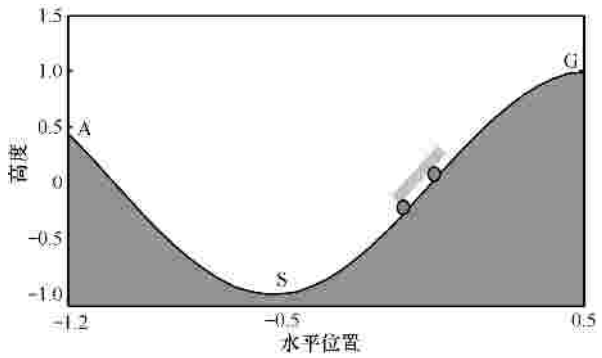


图 5 Mountain Car 示意

图 5 中的曲线代表一个山谷的地形，其中，S 为山谷最低点，G 为右端最高点，A 为左端最高点。小车的任务是在动力不足的条件下，从 S 点以尽量短的时间运动到 G 点。系统的状态由 2 个连续变量 y 和 v 表示，其中，y 为小车的水平位移，v 为小车的水平速度，状态空间约定如式(12)

$$\{\boldsymbol{x} \mid \boldsymbol{x}=[y,v]^{\mathrm{T}}\} \subseteq \mathbb{R}^2,\quad y \in [-1.2, 0.5], v \in [-0.07, 0.07] \tag{12}$$

当小车位于 S 点、G 点和 A 点时，y 的取值分别为- 0.5、0.5 和- 1.2。动作空间为 $\{u_1, u_2, u_3\}$，包含 3 个离散的控制量，即 $u_1=+1$、$u_2=0$ 和 $u_3=-1$，表示小车所受水平方向的力，分别代表全油门向前、零油门和全油门向后 3 个控制行为。仿真实验中，系统的动力学特性描述如式(13)

$$\begin{cases} \dot{v} = \mathrm{bound}[v + 0.001u - g\cos(3y)] \\ \dot{y} = \mathrm{bound}[y + \dot{v}] \end{cases} \tag{13}$$

其中，$g=0.0025$ 为与重力有关的系数，u 为控制量，$\dot{v}$ 为新的水平速度，$\dot{y}$ 为新的水平位移。目标是在没有任何模型先验知识的前提下，控制小车以最短时间从 S 点运动到 G 点。上述控制问题可以用一个确定性 MDP 来建模，奖赏函数如式(14)

$$r_t = \begin{cases} -1, & y < 0.5 \\ 0, & y \geq 0.5 \end{cases} \tag{14}$$

5.2 实验设置

将本文所提 ANRBF-GD-Sarsa(?)算法与 Sutton 等人提出的 GD-Sarsa(?)算法[1]及 Maei 等人提出的 Greedy-GQ 算法[14]在相同的实验环境下分别独立重复做 30 次仿真实验，并比较不同参数值对算法的影响。仿真实验中，每次实验设定的情节数为 1 000，每个情节的最大时间步数设置为 1 000。小车的初始状态设为 $y=-0.5$ 和 $v=0$，当小车到达 G 点或时间步数超过 1 000 时，一个情节结束。然后系统状态重新进行初始化，开始下一个情节的学习。学习完设定的 1 000 个情节后，一次实验结束。学习算法的性能由 3 个指标来评价：①收敛速度，即算法能够在多少个情节内收敛；②收敛结果，即算法收敛后，小车从初始点到达目标点 G 所用的时间步；③初始性能，即每次实验的前 5 个情节中，小车从初始点到达目标点 G 所用的时间步。

本问题的状态空间为一个二维连续空间，首先将状态空间进行高斯划分，在每一维分别用 8 个等距的高斯基函数来对状态空间进行划分，则本问题势函数定义如式(15)

$$\begin{aligned} F(\boldsymbol{x}) &= \sum_{i=1}^{64} w_i f_i(\boldsymbol{x}) \\ f_i(\boldsymbol{x}) &= \varphi_i(\boldsymbol{x}) \left[\sum_{k=1}^{64} \varphi_k(\boldsymbol{x}) \right]^{-1} \\ \varphi_i(\boldsymbol{x}) &= \exp\left(-\sum_{j=1}^{n} \frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2} \right) \end{aligned} \tag{15}$$

其中，$\boldsymbol{c}_i=[c_{i1}, \cdots, c_{in}]^{\mathrm{T}}$ 是宽度为 $\boldsymbol{\sigma}_i=[\sigma_{i1}, \cdots, \sigma_{in}]^{\mathrm{T}}$ 的

高斯函数f_i的中心位置。

针对式(12)所示的二维连续状态空间，使用10个8×8的Tiling来编码，10个Tiling按照随机偏移规则来覆盖状态空间。状态编码过程如下：在设置好的10个Tiling中，查找状态落在哪10个Tile的接受区域，在对应的640维二进制编码中将对应位置置1，其余位置置0。本实验中每个状态经过10个8×8的Tiling编码后，采用Hash机制进行变换，得到对应的特征向量。

实验中状态动作对的特征向量维数设置为3 000维，即$j(x,u)\in i^{3000}$，折扣率$g=1.0$，衰减因子$l=0.9$，权值向量为$?\in i^{3000}$，初始化为0。采用如式（15）所述的8×8个等距的高斯基函数来对二维状态空间进行高斯划分，得到ANRBF网络势函数模型。高斯基函数的宽度向量设为$s=[0.1,0.1]^{\mathrm{T}}$，ANRBF输出层权值向量$w\in i^{8\times8}$，初始化为0，ANRBF学习率为$b=0.05$。另外，考虑到本问题奖赏值设定如式(14)所示，结合问题特性，行为策略使用e-greedy，$e=0$，即贪心策略。由于学习过程中得到的奖赏都是–1，权值向量初值都为0，故初始值函数为0，而最终真实值函数都是负的，所以在学习初期即使$e=0$也保证了有效的探索。随着值函数不断地逼近真实值，此行为策略将指导小车获得最优爬山路径。

5.3　实验分析

ANRBF-GD-Sarsa(?)算法、GD-Sarsa(?)算法及Greedy-GQ算法的30次仿真实验结果如图6所示。图6分3行3列共9幅图，横坐标表示情节数，最大设为1 000，纵坐标为每个情节小车从谷底爬到目标点G所走的时间步数，最大时间步设为1 000。图6 3列对应算法的学习率取值分别为0.05/10=0.005、0.15/10=0.015及0.5/10=0.05，这里的10是状态编码环节所用的Tiling数目；1行、2行、3行分别是Greedy-GQ算法、GD-Sarsa(?)算法及ANRBF-GD-Sarsa(?)算法，其中，每一幅图是某一种算法在对应学习率下的30次仿真实验的学习结果。从图6列向比较中可以看出，在相同学习率下，ANRBF-GD-Sarsa(?)算法初始性能及收敛速度最好，GD-Sarsa(?)算法次之，Greedy-GQ算法最差；从图6行向比较中可以看出，3个算法随着学习率的增大收敛速度明显加快，Greedy-GQ算法反差最为明显，GD-Sarsa(?)算法反差稍小，ANRBF-GD-Sarsa(?)算法反差最小，可见，ANRBF-GD-Sarsa(?)算法对学习率的适应性较强。

上述30次仿真运行结果的均值比较如图7所示。图7从上向下共3幅图，学习率分别为0.005、0.015、0.05，从图中可以直观地看出，在收敛速度和初始性能方面，ANRBF-GD-Sarsa(?)算法相比GD-Sarsa(?)算法及Greedy-GQ算法效果更好，而在最终收敛效果方面，学习率为0.005时，本文所提算法ANRBF-GD-Sarsa(?)与GD-Sarsa(?)算法优于Greedy-GQ算法，而在学习率为0.015及0.05时，3种算法收敛结果基本相同。

图6和图7展示的实验结果详细数据分析见表1~表3。3张表列出了3个算法采用不同学习率独立运行30次仿真实验所得结果的统计值。表1

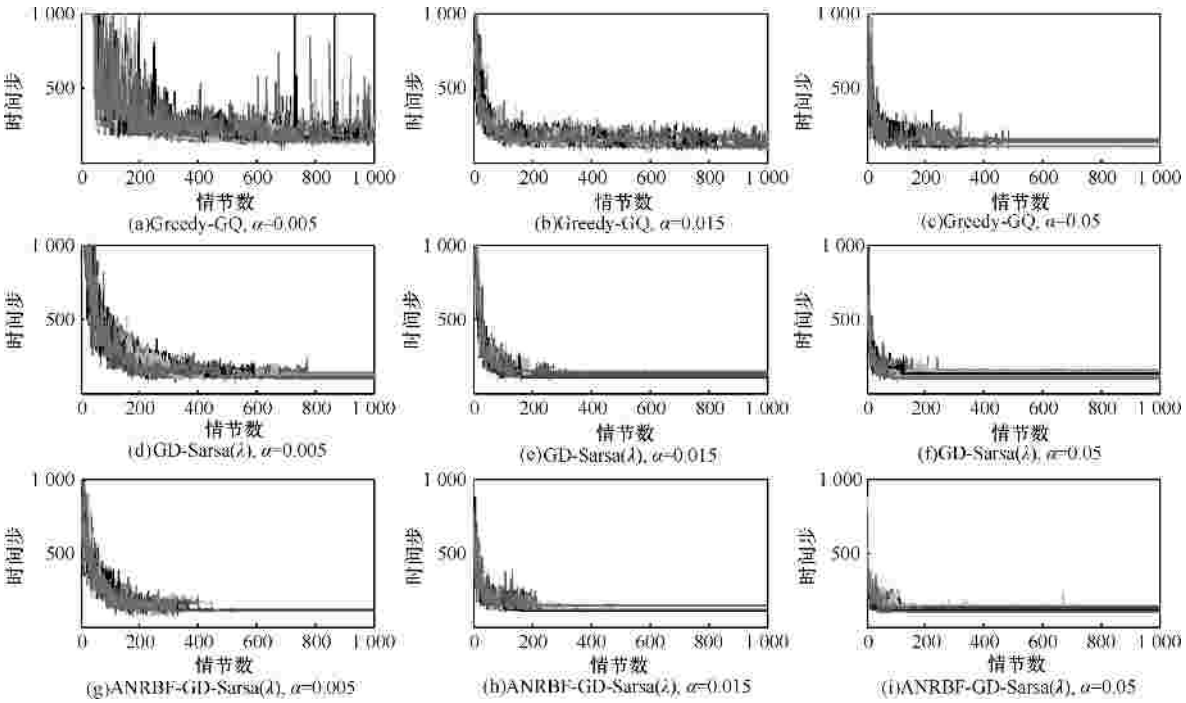


图6　Mountain Car问题中3种算法的性能比较

列出了 30 次实验中算法收敛所需要的情节数的最

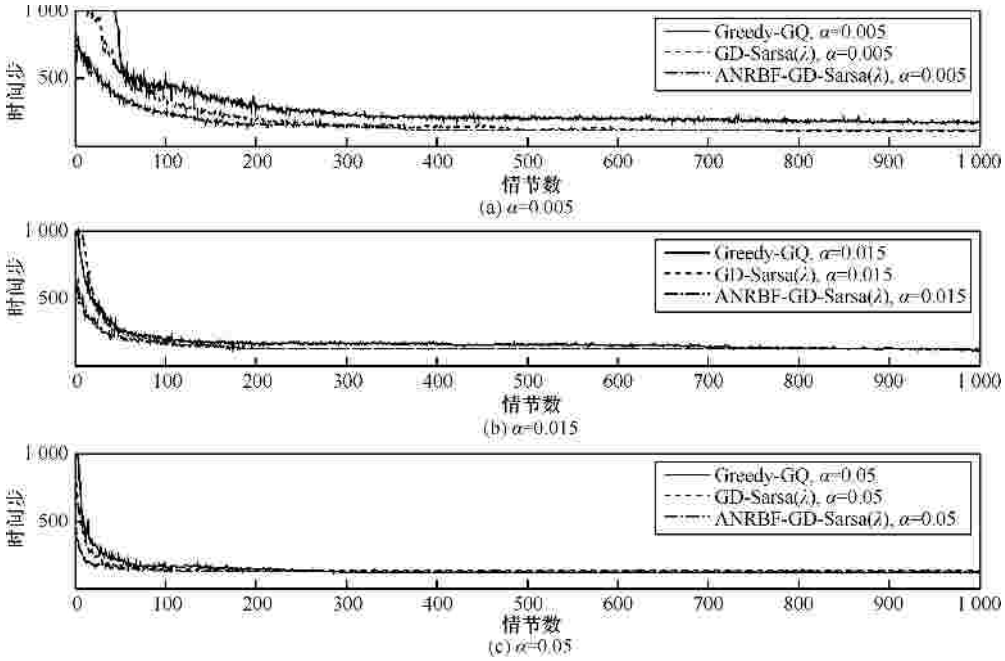


图 7　Mountain Car **问题中** 3 **种算法的性能比较**

表 1　　**3 个算法收敛所需情节数比较**

算法	*a*=0.005			*a*=0.015			*a*=0.05		
	最大	最小	平均	最大	最小	平均	最大	最小	平均
Greedy-GQ	*N*	*N*	*N*	*N*	*N*	*N*	487±50	170±50	357±10
GD-Sarsa(*?*)	773±20	324±20	762±10	272±30	100±30	200±10	130±50	97±50	112±10
ANRBF-GD-Sarsa(*?*)	445±20	300±20	367±10	217±30	109±30	172±10	210±50	103±50	110±10

表 2　　**3 个算法收敛后到达目标点所需时间步比较**

算法	*a*=0.005			*a*=0.015			*a*=0.05		
	最大	最小	平均	最大	最小	平均	最大	最小	平均
Greedy-GQ	*N*	*N*	*N*	*N*	*N*	*N*	157±20	105±20	115±10
GD-Sarsa(*?*)	143±10	105±10	113±5	149±10	106±10	120±5	153±10	108±10	129±5
ANRBF-GD-Sarsa(*?*)	118±10	112±10	113±5	147±10	112±10	123±5	150±10	104±10	125±5

表 3　　**3 个算法前 5 个情节到达目标点所需时间步比较**

算法	*a*=0.005					*a*=0.015					*a*=0.05				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Greedy-GQ	1 000	1 000	1 000	1 000	1 000	965	967	994	929	881	1 000	1 000	978	886	798
GD-Sarsa(*?*)	1 000	1 000	1 000	1 000	1 000	1 000	1 000	1 000	986	995	789	770	623	554	612
ANRBF-GD-Sarsa(*?*)	722	671	791	725	694	633	561	619	529	465	498	357	366	314	296

大、最小及平均值；表 2 **列出了** 30 **次实验中算法收敛后，小车从谷底爬到山顶** G **所需的时间步数的最大、最小及平均值；表** 3 **重点比较** 3 **个算法的初始性能，关注** 30 **次仿真实验中的前** 5 **个情节时间步数的平均值。因为结果曲线存在一定的随机和震荡性，所以这里最大、最小和平均值统一用一个区间表示。当学习率为** 0.005 **及** 0.015 **时，**Greedy-GQ **算法在设定的** 1 000 **个情节内并未收敛，在图** 6 **中反映为曲线图震荡比较厉害，在表中标识为** *N*。**从表** 1 **和表** 2 **中可以看出，学习率为** 0.005 **时，**

ANRBF-GD-Sarsa(?)算法可以在大约367个情节收敛，收敛后大约113个时间步到达目标点。GD-Sarsa(?)算法需要大约762个情节收敛，收敛后大约113个时间步到达目标点。Greedy-GQ算法没有收敛；学习率为0.015时，ANRBF-GD-Sarsa(?)算法可以在大约172个情节收敛，收敛后大约123个时间步到达目标点。GD-Sarsa(?)算法需要大约200个情节收敛，收敛后大约120个时间步到达目标点。Greedy-GQ算法没有收敛；学习率为0.05时，ANRBF-GD-Sarsa(?)算法可以在大约110个情节收敛，收敛后大约125个时间步到达目标点。GD-Sarsa(?)算法需要大约112个情节收敛，收敛后大约129个时间步到达目标点。Greedy-GQ算法需要大约357个情节收敛，收敛后大约115个时间步到达目标点。从表3中可以看出，30次仿真实验平均结果中，比较算法前5个情节的时间步，ANRBF-GD-Sarsa(?)最优。综上所述，3个算法收敛后到达目标点时间步数基本相同，而在初始性能及收敛速度方面，ANRBF-GD-Sarsa(?)明显优于GD-Sarsa(?)算法及Greedy-GQ算法。

6 结束语

本文旨在通过势函数塑造奖赏机制来提高强化学习算法的初始性能及收敛速度。创新之处在于利用ANRBF网络将输入空间映射到高维特征空间，通过一组高斯基函数的线性组合构建势函数。在一定程度上利用了环境模型的先验知识，从而可以有效提高算法初始性能及收敛速度。基于ANRBF网络和线性梯度下降方法提出了ANRBF-GD-Sarsa(?)算法。从理论上分析了ANRBF-GD-Sarsa(?)算法的收敛性，并通过连续状态空间、离散动作空间的Mountain Car仿真问题对算法的有效性进行了验证。与GD-Sarsa(?)算法及Greedy-GQ算法进行了比较，结果表明，本文所提算法具有更优的性能。

函数逼近为解决连续状态空间强化学习问题提供了一个有效的途径。近来利用各种有监督学习方法来逼近状态（动作）值函数成为强化学习领域的研究热点，函数逼近可以大大提高算法的学习速度，但是也面临算法复杂化及理论上难以保证收敛等问题，如何在保证算法的收敛性下有效提高算法的收敛速度，是未来强化学习函数逼近研究的热点。如何设计更好的势函数，以及势函数如何自学习，从而更好地将模型知识传递给学习器，以提高算法初始性能及收敛速度也是研究的热点。

参考文献：

[1] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[M]. Cambridge: MIT Press, 1998.163-223.

[2] 刘全, 傅启明, 龚声蓉等. 最小状态变元平均奖赏的强化学习方法[J]. 通信学报, 2011, 32(1): 66-71.
LIU Q, FU Q M, GONG S R, *et al*. Reinforcement learning algorithm based on minimum state method and average reward[J]. Journal on Communications[J], 2011, 32(1): 66-71.

[3] SILVER D, SUTTON R S, MÜLLER M. Temporal-difference search in computer go[J]. Machine Learning, 2012, 87(2): 183-219.

[4] 陈宗海, 文锋, 聂建斌等. 基于节点生长 k-均值聚类算法的强化学习方法[J]. 计算机研究与发展, 2006, 34(4): 661-666.
CHEN Z H, WEN F, NIE J B, *et al*. A reinforcement learning method based on node-growing k-means clustering algorithm[J]. Journal of Computer Research and Development, 2006, 34(4): 661-666.

[5] 刘全, 闫其粹, 伏玉琛等. 一种基于启发式奖赏函数的分层强化学习方法[J]. 计算机研究与发展, 2011, 48(12): 2352-2358.
LIU Q, YAN Q C, FU Y C, *et al*. A hierarchical reinforcement learning method based on heuristic reward function[J]. Journal of Computer Research and Development, 2011, 48(12): 2352-2358.

[6] KRETCHMAR R M. Parallel reinforcement learning[A]. World Conference on Systemics, Cybernetics, and Informatics[C]. New York, 2002. 114-118.

[7] GEIST M, PIETQUIN O. Parametric value function approximation: a unified view[A]. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning[C]. Paris, 2011. 9-16.

[8] 王雪松, 田西兰, 程玉虎等. 基于协同最小二乘支持向量机的Q学习[J]. 自动化学报, 2009, 35(2): 214-219.
WANG X S, TIAN X L, CHENG Y H, *et al*. Q-learning system based on cooperative least squares support vector machine[J]. Acta Automatica Sinica, 2009, 35(2): 214-219.

[9] TSITSIKLIS J N, VAN ROY B. An analysis of temporal-difference learning with function approximation[J]. IEEE Transactions on Automatic Control, 1997, 42: 674-690.

[10] SUTTON R S, Mcallester D, SINGH S, *et al*. Policy gradient methods for reinforcement learning with function approximation[A]. Annual Conference on Neural Information Processing Systems[C]. Denver, 1999. 1057-1063.

[11] BONARINI A, LAZARIC A, MONTRONE F, *et al*. Reinforcement distribution in fuzzy Q-learning[J]. Fuzzy Sets and Systems, 2009, 160(10): 1420-1443.

[12] HEINEN M R, ENGEL P M. An incremental probabilistic neural network for regression and reinforcement learning tasks[A]. International Conference on Artificial Neural Networks[C]. Thessaloniki, 2010. 170-179.

[13] MAEI H R, SUTTON R S. GQ(?): a general gradient algorithm for temporal difference prediction learning with eligibility traces[A]. International Conference on Artificial General Intelligence[C]. Lugano,

2010. 91-96.

[14] MAEI H R, SZEPESVARI C, BHATNAGAR S, *et al*. Toward off-policy learning control with function approximation[A]. International Conference on Machine Learning[C]. Haifa, 2010. 719-726.

[15] RANDLØV J, ALSTRØM P. Learning to drive a bicycle using reinforcement learning and shaping[A]. International Conference on Machine Learning[C]. San Francisco, 1998. 463-471.

[16] NG A Y, HARADA D, RUSSELL S. Policy invariance under reward transformations: theory and application to reward shaping[A]. International Conference on Machine Learning[C]. San Francisco, 1999. 278-287.

[17] HUANG G, SARATCHANDRAN P, SUNDARARAJAN N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation[J]. IEEE Transactions on Neural Networks, 2005, 16(1): 57-67.

[18] SHERSTOV A A, STONE P. Function approximation via tile coding: automating parameter choice[A]. International Symposium on Abstraction, Reformulation and Approximation[C]. Berlin, 2005. 194-205.

作者简介：

肖飞（1988-），男，江苏沭阳人，苏州大学硕士生，主要研究方向为强化学习。

刘全（1969-），男，内蒙古牙克石人，苏州大学教授、博士生导师，主要研究方向为强化学习、智能信息处理和自动推理。

傅启明（1985-），男，江苏淮安人，苏州大学博士生，主要研究方向为强化学习、贝叶斯推理和遗传算法。

孙洪坤（1988-），男，江苏淮安人，苏州大学硕士生，主要研究方向为强化学习。

高龙（1988-），男，江苏盐城人，苏州大学硕士生，主要研究方向为贝叶斯推理。