文章编号: 1001-0920(2017)12-2153-09 DOI: 10.13195/j.kzyjc.2016.1148

基于状态聚类的非参数化近似广义策略迭代增强学习算法

季 挺, 张 华†

(南昌大学 江西省机器人与焊接自动化重点实验室，南昌 330031)

摘 要: 为解决当前近似策略迭代增强学习算法普遍存在计算量大、基函数不能完全自动构建的问题,提出一种基于状态聚类的非参数化近似广义策略迭代增强学习算法(NPAGPI-SC). 该算法利用二级随机采样过程采集样本,利用trial-and-error过程和以样本完全覆盖为目标的估计方法计算逼近器初始参数,利用delta规则和最近邻思想在学习过程中自适应地调整逼近器,利用贪心策略选择应执行的动作. 一级倒立摆平衡控制的仿真实验结果验证了所提出算法的有效性和鲁棒性.

关键词: 增强学习；策略迭代；非参数化；状态聚类

中图分类号: TP181 **文献标志码:** A

Nonparametric approximation generalized policy iteration reinforcement learning algorithm based on states clustering

JI Ting, ZHANG Hua†

(Key Lab of Robot & Welding Automation of Jiangxi Province，Nanchang University，Nanchang 330031，China)

Abstract: A nonparametric approximation generalized policy iteration reinforcement learning algorithm based on states clustering(NPAGPI-SC) is proposed to solve the problems such as large calculating quantity and building basis function incompletely automated for the current approximation policy iteration reinforcement learning algorithm. In this algorithm, two stage random sampling process is used to collect samples, the trial-and-error process and the estimation algorithm for covering samples completely are utilized to compute approximator's initial parameters, the delta rule and nearest neighbor method are exploited to adjust the approximator automatically in the learning process, and the greedy strategy is adopted to select an action. The results of simulation on the balancing control of a single inverted pendulum show the effectiveness and robustness of the proposed algorithm.

Keywords: reinforcement learning；policy iteration；nonparametric；states clustering

0 引 言

经典增强学习算法使用表格存储和计算Q值(或V值),仅适于解决小规模、离散状态和动作空间中的问题,实际系统通常工作在大规模、连续状态和动作空间中,若依然采取表格算法则会导致“维数灾”问题. 近似策略迭代算法是解决“维数灾”问题的主要方法之一,典型的近似策略迭代增强学习算法LSPI[1]为离线算法,不仅计算量大,而且值函数逼近器(包括基函数和参数)依赖先验知识或通过反复试凑确定,不具备自动构建的能力. online LSPI[2]解决了LSPI算法的在线计算问题,BLSPI[3]在online LSPI算法的基础上进一步提高了样本的利用率,并在一定程度上降低了近似策略迭代算法的计算量,但这两种算法依然需要手工构建基函数. 基于核的近似策略迭代增强学习算法[4-7]能够在相当程度上解决逼近器的自动构建问题,但依然需要手动选择基函数宽度、稀疏度阈值等参数.

鉴于此,本文提出一种基于状态聚类的非参数化近似广义策略迭代增强学习算法(NPAGPI-SC). 该算法能够综合利用离线和在线样本,不仅计算量较小,而且只需预先指定增强学习的允许误差率,便能够在与环境的交互过程中自主构建和调整逼近器的基函数和参数. 将NPAGPI-SC应用于一级倒立摆平衡控制问题,仿真实验结果验证了所提出算法的有效性和鲁棒性,且相较于LSPI、online LSPI、BLSPI、KLSPI[5-6]等算法,NPAGPI-SC具有收敛速度更快的

收稿日期: 2016-09-09；**修回日期:** 2016-12-13.

基金项目: 国家863计划项目(SS2013AA041003).

作者简介: 季挺(1982–), 男, 博士, 从事智能机器人、智能控制的研究；张华(1964–), 男, 教授, 博士生导师, 从事智能机器人技术、光纤传感、智能金属结构等研究.

†通讯作者. E-mail: zhanghua@163.com

优势.

1 NPAGPI-SC算法网络结构

NPAGPI-SC算法网络结构由状态输入层、Q值函数逼近器和动作选择器3部分组成,如图1所示.

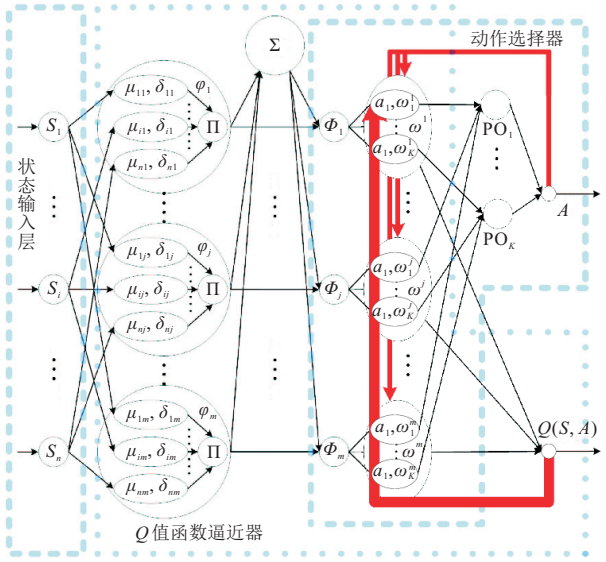

图 1 NPAGPI-SC算法网络结构

1.1 状态输入层

状态输入层接收输入状态$S=(s_1,\cdots,s_i,\cdots,s_n)^{\mathrm{T}}\in R^n$. 其中:$n$为状态空间的维数,$s_i$为$S$在各维空间上的分量.

1.2 Q值函数逼近器

Q值函数逼近器由基于RBF的线性逼近结构实现,状态动作对(S,A)对应的近似Q值计算公式为

$$Q(S,A)=\Phi^{\mathrm{T}}(S)\omega(A). \tag{1}$$

$\Phi(S)=(\Phi_1(S),\cdots,\Phi_j(S),\cdots,\Phi_m(S))^{\mathrm{T}}$为状态$S$在各基函数下的归一化隶属度向量,定义为

$$\Phi_j(S)=\frac{\varphi_j(S)}{\sum\limits_{l=1}^{m}\varphi_l(S)},\ j=1,2,\cdots,m. \tag{2}$$

其中:$\varphi(S)=(\varphi_1(S),\cdots,\varphi_j(S),\cdots,\varphi_m(S))^{\mathrm{T}}$为逼近器的状态基函数向量,其值为状态$S$在各基函数下的隶属度;$m$为$\varphi(S)$的维数. $\varphi(S)$使用RBF函数定义为

$$\varphi_j(S)=\prod_{i=1}^{n}\exp\left[-\frac{(s_i-\mu_{ij})^2}{2\delta_{ij}^2}\right],\ j=1,2,\cdots,m. \tag{3}$$

其中:$\mu_j=(\mu_{1j},\cdots,\mu_{ij},\cdots,\mu_{nj})^{\mathrm{T}}$和$\delta_j=(\delta_{1j},\cdots,\delta_{ij},\cdots,\delta_{nj})^{\mathrm{T}}$分别为基函数$\varphi_j$的中心和半径,$\mu_{ij}$和$\delta_{ij}$分别为基函数$\varphi_j$在第$i$维状态分量上的中心和半径.

$\omega=(\omega^1,\cdots,\omega^j,\cdots,\omega^m)^{\mathrm{T}}$为逼近器的插值参数向量,其意义为在各状态基函数中心执行所选动作A的Q值,利用插值方法[8]定义为

$$\omega^j=\frac{\sum\limits_{k=1}^{K}d(a_k,A)\cdot\omega_k^j}{\sum\limits_{k=1}^{K}d(a_k,A)},\ j=1,2,\cdots,m. \tag{4}$$

$$d(a_k,A)=\exp\left(-\frac{(a_k-A)^2}{2\xi^2}\right),$$

$$\xi=\frac{\sum\limits_{k=1}^{K}|a_k-A|}{K}.$$

其中:K为可选离散动作的数量;a_k为第k个可选的离散动作;ω_k^j为逼近器参数,其意义是在状态基函数φ_j中心μ_j执行动作a_k的投票数,采用delta规则进行更新,定义为

$$\omega_k^j(t+1)=\omega_k^j(t)+\eta\Delta\Phi_j(S_t)\frac{d(a_k,A)}{\sum\limits_{k=1}^{K}d(a_k,A)}. \tag{5}$$

这里:η为Q值函数逼近器平均学习率,由初始状态基函数构建过程确定;Δ为TD误差,定义为

$$\Delta=r+\gamma\max_{A_{t+1}}Q(S_{t+1},A_{t+1})-Q(S_t,A_t), \tag{6}$$

r为即时奖励,γ为折扣率.

状态基函数向量φ(包括其参数和维数)与插值参数向量ω的维数由Q值函数逼近器自动构建过程(见第2节)依据采样样本确定,并在学习过程中自适应调整.

1.3 动作选择器

动作选择器采用贪心策略实现,计算公式为

$$A=\arg\max_{a_k}(\mathrm{PO}_1,\cdots,\mathrm{PO}_k,\cdots,\mathrm{PO}_K), \tag{7}$$

其中PO_k是对当前状态S下执行动作a_k的投票结果,定义为

$$\mathrm{PO}_k=\sum_{j=1}^{m}\Phi_j(S)\omega_k^j, \tag{8}$$

$$E(\mathrm{PO}_k)=\sum_{j=1}^{m}\Phi_j(S)E(\omega_k^j), \tag{9}$$

$$\mathrm{Var}(\mathrm{PO}_k)=\sum_{j=1}^{m}\Phi_j^2(S)\mathrm{Var}(\omega_k^j). \tag{10}$$

由式(9)和(10)可以看出,投票器算法可有效降低单个ω_k^j估计误差对动作选择结果的影响,能够大大提升算法对动作噪声的抗干扰能力.

2 Q值函数逼近器自动构建过程

Q值函数逼近器自动构建过程由二级随机采样过程、核心状态聚类生成过程、初始状态基函数构建

过程组成.

2.1　二级随机采样过程

近似增强学习方法中的近似不仅指表示近似,也指样本近似. 在大规模、连续状态空间中遍历每个状态是不可能的,因此需要一种好的采样方法尽量保证样本空间对状态空间的覆盖率. 本文提出一种二级随机采样过程,该过程分为局部随机采样过程和全局随机采样过程两个层次,依据预先指定的增强学习允许误差率ε自动收集状态样本.

2.1.1　局部随机采样过程

局部随机采样过程如下.

Step 1: 采用随机策略运行一次目标系统. 设置采样次数$\mathrm{tm}=1$,目标系统运行次数$\mathrm{tr}=1$,采样缓存长度$L=\mathrm{step}$,step为目标系统本次运行持续的时间步数(同时也是目标系统本次运行采集样本的数量);将本次采样结果记为$\mathrm{sam}_{\mathrm{tm}}$,通过下式分别计算均值$\mathrm{mean}_{\mathrm{tm}}$及其在各维分量上的平均绝对离差向量$\mathrm{mad}_{\mathrm{tm}}$:

$$\mathrm{mean}_{\mathrm{tm}}=\frac{\sum_{h=1}^{\mathrm{ns}}(\mathrm{sam}_{\mathrm{tm}})_h}{\mathrm{ns}},\tag{11}$$

$$\mathrm{mad}_{\mathrm{tm}}=\frac{\sum_{h=1}^{\mathrm{ns}}\mathrm{abs}((\mathrm{sam}_{\mathrm{tm}})_h-\mathrm{mean}_{\mathrm{tm}})}{\mathrm{ns}},\tag{12}$$

其中ns为$\mathrm{sam}_{\mathrm{tm}}$中样本的数量.

Step 2: 开始单次采样. $\mathrm{tm}=\mathrm{tm}+1$,初始化单次采样新增非重复样本数量$\mathrm{ln}=0$,使$\mathrm{temp}=\mathrm{sam}_{\mathrm{tm}-1}$.

Step 3: 采用随机策略运行一次目标系统. $\mathrm{tr}=\mathrm{tr}+1$,收集本次运行采集到的样本集合$T_{\mathrm{sam}}$,同时更新step和rep的值,rep为$T_{\mathrm{sam}}$与temp中重复样本的数量.

Step 4: 使得$\mathrm{temp}=\mathrm{temp}\bigcup T_{\mathrm{sam}}$,更新ln与$L$的值,有

$$\mathrm{ln}=\mathrm{ln}+\mathrm{step}-\mathrm{rep}.\tag{13}$$

$$L=\begin{cases}L-1,\ \mathrm{rep}==\mathrm{step};\\ \mathrm{round}\Big(\dfrac{L(\mathrm{tr}-1)+\mathrm{step}}{\mathrm{tr}}\Big),\ \mathrm{otherwise}.\end{cases}\tag{14}$$

若$\mathrm{ln}<L$,则执行Step 3,否则,令$\mathrm{sam}_{\mathrm{tm}}=\mathrm{temp}$,执行Step 5.

Step 5: 由式(11)和(12)计算$\mathrm{mean}_{\mathrm{tm}}$和$\mathrm{mad}_{\mathrm{tm}}$.

Step 6: 计算$\mathrm{sam}_{\mathrm{tm}}$相对$\mathrm{sam}_{\mathrm{tm}-1}$的变化率rate,有

$$\mathrm{rate}=\|(\mathrm{rate}_1,\cdots,\mathrm{rate}_i,\cdots,\mathrm{rate}_n)\|_2.\tag{15}$$

$$\mathrm{rate}_i=\begin{cases}1,\ \mathrm{bc}_i\leqslant\mathrm{fl}_i|\mathrm{bl}_i\leqslant\mathrm{fc}_i;\\ \dfrac{|\mathrm{fc}_i-\mathrm{fl}_i|+|\mathrm{bc}_i-\mathrm{bl}_i|}{\max(\mathrm{fbcl})-\min(\mathrm{fbcl})},\ \mathrm{otherwise}.\end{cases}$$

$$\mathrm{fbcl}=\{\mathrm{fc}_i,\mathrm{bc}_i,\mathrm{fl}_i,\mathrm{bl}_i\}.$$

$$\mathrm{fc}=\mathrm{mean}_{\mathrm{tm}}-\mathrm{mad}_{\mathrm{tm}}.$$

$$\mathrm{bc}=\mathrm{mean}_{\mathrm{tm}}+\mathrm{mad}_{\mathrm{tm}}.$$

$$\mathrm{fl}=\mathrm{mean}_{\mathrm{tm}-1}-\mathrm{mad}_{\mathrm{tm}-1}.$$

$$\mathrm{bl}=\mathrm{mean}_{\mathrm{tm}-1}+\mathrm{mad}_{\mathrm{tm}-1}.$$

Step 7: 若$\mathrm{rate}<\varepsilon$,则局部采样过程结束,否则执行Step 2.

2.1.2　全局随机采样过程

全局随机采样过程如下.

Step 1: 目标系统运行一次局部随机采样过程. 设置局部采样次数$\mathrm{TM}=1$,将本次局部采样结果记为$\mathrm{SAM}_{\mathrm{TM}}$,类似式(11)和(12)分别计算均值$\mathrm{MEAN}_{\mathrm{TM}}$及其在各维分量上的平均绝对离差向量$\mathrm{MAD}_{\mathrm{TM}}$.

Step 2: 目标系统运行一次局部随机采样过程,得到本次局部采样结果T_{SAM}. $\mathrm{TM}=\mathrm{TM}+1$,$\mathrm{SAM}_{\mathrm{TM}}=\mathrm{SAM}_{\mathrm{TM}-1}\bigcup T_{\mathrm{SAM}}$.

Step 3: 类似式(11)和(12)计算$\mathrm{MEAN}_{\mathrm{TM}}$和$\mathrm{MAD}_{\mathrm{TM}}$.

Step 4: 类似式(15)计算$\mathrm{SAM}_{\mathrm{TM}}$相对$\mathrm{SAM}_{\mathrm{TM}-1}$的变化率RATE.

Step 5: 若$\mathrm{RATE}<\varepsilon$,最终样本集$\mathrm{SAM}=\mathrm{SAM}_{\mathrm{TM}}$,则全局采样过程结束,否则执行Step 2.

2.2　核心状态聚类生成过程

核心状态聚类生成过程的目的是为初始状态基函数构建过程提供初值,包括trial-and-error过程和总体生成过程两个层次.

2.2.1　trial-and-error过程

trail-and-error过程[9]是一种使用相对准则判定样本集最佳聚类数的方法,本文基于轮廓指标[10]、采用K均值聚类算法实现该过程,具体方法如下.

Step 1: 初始化输入样本集sam的可能聚类结果集合$\mathrm{clus}=\varnothing$、clus对应的有效性集合$\mathrm{val}=\varnothing$;使用式(16)初始化sam的可能聚类数集合num, num的最大取值为$\sqrt{N}$[11], N为sam中样本的数量. 设置每种聚类数下运行K均值聚类算法的次数RTM(一般$3\sim10$次),有

$$\mathrm{num}=\begin{cases}\{1,2,\cdots,[\sqrt{N}]\},\ \mathrm{num\ is}\ \varnothing;\\ \mathrm{num},\ \mathrm{otherwise}.\end{cases}\tag{16}$$

Step 2: 遍历num,对当前遍历的聚类数num_v运

行 K 均值聚类算法RTM次,得到聚类结果 $(\mathrm{TC}_1,\cdots,\mathrm{TC}_w,\cdots,\mathrm{TC}_{\mathrm{RTM}})$ 及其相应的聚类有效性 $(\mathrm{TV}_1,\cdots,\mathrm{TV}_w,\cdots,\mathrm{TV}_{\mathrm{RTM}})$,$\mathrm{TV}_w$ 采用轮廓指标[10]计算,有

$$\mathrm{TV}_w=\frac{1}{\mathrm{num}_v}\sum_{p=1}^{\mathrm{num}_v}\frac{1}{n_p}\sum_{q:x_q\in(\mathrm{TC}_w)_p}\frac{b_q-a_q}{\max(b_q,a_q)}. \tag{17}$$

$$a_q=d_{\mathrm{avg}}^{\mathrm{ps}}(x_q,(\mathrm{TC}_w)_p-\{x_q\}),$$

$$b_q=\min_{u=1,2,\cdots,\mathrm{num}_v,u\neq p}d_{\mathrm{avg}}^{\mathrm{ps}}(x_q,(\mathrm{TC}_w)_u).$$

其中:n_p 为聚类 $(\mathrm{TC}_w)_p$ 的基数,a_q 为 $(\mathrm{TC}_w)_p$ 中元素 x_q 到 $(\mathrm{TC}_w)_p$ 中其他元素的平均距离,b_q 为 x_q 到除 $(\mathrm{TC}_w)_p$ 之外最接近 x_q 的聚类的平均距离,$d_{\mathrm{avg}}^{\mathrm{ps}}(\cdot,\cdot)$ 为一个点到一个集合的平均距离测度.

Step 3:将 num_v 下的最优聚类及其有效性存入 clus_v 和 val_v,有

$$id=\arg\max_w(\mathrm{TV}_1,\cdots,\mathrm{TV}_w,\cdots,\mathrm{TV}_{\mathrm{RTM}}),$$

$$\mathrm{clus}_v=\mathrm{TC}_{id}; \tag{18}$$

$$\mathrm{val}_v=\frac{\sum_{w=1}^{\mathrm{RTM}}\mathrm{TV}_w}{\mathrm{RTM}}. \tag{19}$$

Step 4:若num遍历结束,则输出sam的最佳聚类数Num和聚类结果clu,trial-and-error过程结束,否则执行Step 2,有

$$\mathrm{ID}=\arg\max_v(\mathrm{val}),$$

$$\mathrm{Num}=\mathrm{num}_{\mathrm{ID}}; \tag{20}$$

$$\mathrm{clu}=\mathrm{clus}_{\mathrm{ID}}. \tag{21}$$

2.2.2 总体生成过程

核心状态聚类总体生成过程如下.

Step 1:初始化全局样本集SAM的可能聚类数集合 $\mathrm{cnum}=\varnothing$.

Step 2:目标系统运行一次局部随机采样过程,得到样本集 T_{SAM}.

Step 3:在 T_{SAM} 上运行trial-and-error过程,获取其最佳聚类数NUM,并更新cnum,有

$$\mathrm{cnum}=\mathrm{cnum}\bigcup\mathrm{NUM}. \tag{22}$$

Step 4:若全局采样过程结束,则初始化SAM的可能聚类数集合为cnum,在SAM上运行trial-and-error过程,获取其最佳聚类结果CLU,并将CLU作为核心状态聚类,否则执行Step 2.

Step 5:计算CLU中每个聚类的中心和半径,并将其存入核心聚类中心矩阵cen和半径矩阵rad. CLU中单个聚类 C 的中心 C_c 为 C 中所有元素的均值,半径 C_r 为 C 中每个元素与其中心 C_c 在各维分量上的距离均值向量,分别为

$$C_c=\frac{\sum_{z=1}^{\mathrm{nc}}C_z}{\mathrm{nc}}, \tag{23}$$

$$C_r=\frac{\sum_{z=1}^{\mathrm{nc}}\mathrm{abs}(C_z-C_c)}{\mathrm{nc}}. \tag{24}$$

其中:C_z 为聚类 C 中的一个元素,nc为聚类 C 中元素的数量.

2.3 初始状态基函数构建过程

状态基函数如式(3)所示. 为了提升状态基函数对样本空间的覆盖率、使其尽可能完整地反映状态空间的分布,本文提出一种以样本空间完全覆盖为目标的估计方法,通过采集到的样本计算逼近器的状态基函数和平均学习率,具体过程如下.

Step 1:初始化基函数向量 φ.

Step 2:遍历样本集SAM. 通过下式判定与当前遍历样本 C_{sam} 最邻近的基函数 φ_{near}:

$$\varphi_{\mathrm{near}}=\arg\min_{\varphi_j}\{||C_{\mathrm{sam}}-\mu_j||j=1,2,\cdots,m\}; \tag{25}$$

通过下式判定 C_{sam} 是否落在 φ_{near} 内部:

$$\mathrm{sum}(\mathrm{abs}(C_{\mathrm{sam}}-\mu_{\mathrm{near}})\leqslant\delta_{\mathrm{near}})==n. \tag{26}$$

若是,则继续遍历过程Step 2,否则结束本次遍历,执行Step 3. 若SAM中所有样本均落在最邻近基函数内部,则表明状态基函数完成了对样本空间的完全覆盖,执行Step 5. 其中,μ_{near} 和 δ_{near} 分别为 φ_{near} 的中心和半径.

Step 3:新增一个状态基函数,将其中心初始化为 C_{sam},并加入 φ 中.

Step 4:通过式(27)和(28)调整 φ 中所有基函数的中心和半径,执行Step 2,有

$$\mu_j=\mu_j+\frac{\sum_{h=1}^{\mathrm{NS}}\mathrm{be}_{hj}\cdot(\mathrm{SAM}_h-\mu_j)}{\sum_{h=1}^{\mathrm{NS}}\mathrm{be}_{hj}}. \tag{27}$$

$$\delta_j=\frac{\sum_{h=1}^{\mathrm{NS}}\mathrm{be}_{hj}\cdot|\mathrm{SAM}_h-\mu_j|}{\sum_{h=1}^{\mathrm{NS}}\mathrm{be}_{hj}}. \tag{28}$$

$$\mathrm{be}_{hj}=\begin{cases}1,\ \mathrm{SAM}_h==\mu_j;\\ \dfrac{\dfrac{1}{|\mathrm{SAM}_h-\mu_j|}}{\displaystyle\sum_{l=1}^{m}\dfrac{1}{|\mathrm{SAM}_h-\mu_l|}},\ \text{otherwise.}\end{cases}$$

其中: NS为SAM中样本的数量, be_{hj} 为样本 SAM_h 到状态基 φ_j 的归一化隶属度.

Step 5: 通过式(29)计算逼近器的平均学习率 η, 初始状态基函数构建过程结束, 有

$$\eta=\frac{2}{\bar{X}+M_e(X)}. \tag{29}$$

其中 $X=(X_1,\cdots,X_j,\cdots,X_m)^{\mathrm{T}}$ 为各基函数的构建样本数向量, 表示为

$$X_j=\sum_{h=1}^{\mathrm{NS}}\mathrm{be}_{hj},\ j=1,2,\cdots,m.$$

3　Q值函数逼近器自适应调整过程

Q值函数逼近器自适应调整过程包括逼近器状态基函数和逼近器参数的自适应调整. 其中, 参数的自适应调整如式(5)所示, 状态基函数的自适应调整过程如下所示.

Step 1: 若输入状态 S 属于状态基函数的构建样本集SAM, 则继续执行Step 1, 判定下一输入状态, 否则将 S 加入构建样本集SAM, 执行Step 2.

Step 2: 通过式(30)搜索与 S 最邻近的基函数 φ_{near}, 有

$$\varphi_{\mathrm{near}}=\arg\lim_{\varphi_j}\{S-\mu_j||j=1,2,\cdots,m\}. \tag{30}$$

Step 3: 若 φ_{near} 为初始状态基函数, TD误差率 $R_\Delta>\varepsilon$, 通过下式判定 S 落在 φ_{near} 外部:

$$R_\Delta=\frac{|\Delta|}{\max(\mathrm{abs}(r))}, \tag{31}$$

$$\mathrm{sum}(\mathrm{abs}(S-\mu_j)\leqslant\delta_j)==n, \tag{32}$$

则新增状态基函数 φ_{new}, 并初始化其中心 $\mu_{\mathrm{new}}=S$, 半径 $\delta_{\mathrm{new}}=\delta_{\mathrm{near}}$, 执行Step 4, 否则直接执行Step 4.

Step 4: 采用delta规则调整状态基函数的中心和半径, 对于 φ_j 有

$$\mu_j=\mu_j+\eta\cdot R_\Delta\cdot\mathrm{BE}_j\cdot(S-\mu_j), \tag{33}$$

$$\delta_j=\delta_j+\eta\cdot R_\Delta\cdot\mathrm{BE}_j\cdot(|S-\mu_j|-\delta_j). \tag{34}$$

其中

$$\mathrm{BE}_j=\frac{\exp\left[-\dfrac{(S-\mu_j)^2}{2\delta_j^2}\right]}{\displaystyle\sum_{l=1}^{m}\exp\left[-\frac{(S-\mu_l)^2}{2\delta_l^2}\right]}.$$

本次调整结束后执行Step 1, 开始下一次状态基函数调整过程.

4　NPAGPI-SC算法步骤

NPAGPI-SC算法步骤分为算法网络结构构建和自主学习两个阶段.

4.1　算法网络结构构建阶段

算法网络结构构建阶段如图2所示, 具体步骤如下所示.

Step 1: 指定增强学习的允许误差率 ε.

Step 2: 使用全局和局部二级随机采样过程采集误差低于 ε 的状态样本集合SAM. 在每次局部采样结束后利用trail-and-error过程计算局部样本集合TSAM的最佳聚类数NUM, 并使用式(22)得到SAM的可能最佳聚类数集合cnum.

Step 3: 依据cnum, 使用trail-and-error过程计算SAM的核心状态聚类CLU. 使用式(23)和(24)分别计算CLU的中心矩阵cen和半径矩阵rad.

Step 4: 分别利用cen和rad初始化基函数向量 φ 的中心和半径, 使用初始状态基函数构建过程计算 φ 和逼近器平均学习率 η, 将 Q 值函数逼近器参数向量 $(\omega_1^1,\cdots,\omega_K^1,\cdots,\omega_1^j,\cdots,\omega_K^j,\cdots,\omega_1^m,\cdots,\omega_K^m)^{\mathrm{T}}$ 初始化为0向量.

Step 5: 依据 φ、η、逼近器初始参数向量构建NPAGPI-SC算法的网络结构(如图2所示).

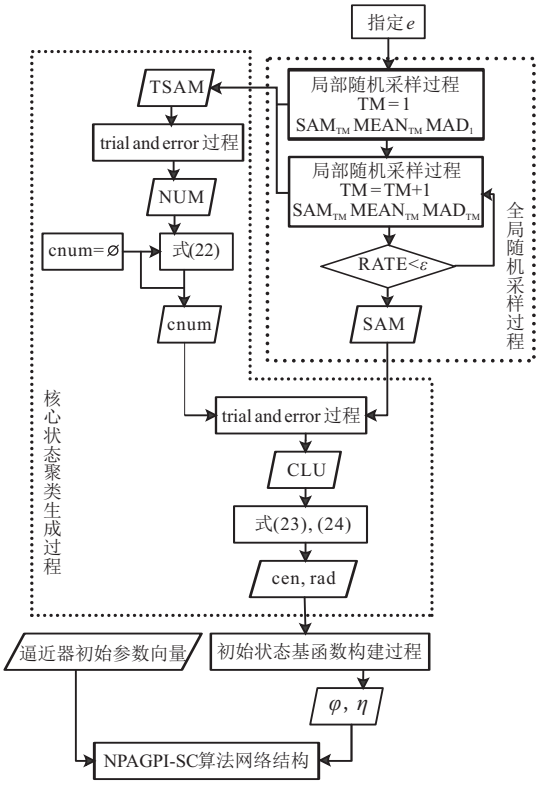

图 2　NPAGPI-SC算法网络结构构建阶段

4.2　自主学习阶段

NPAGPI-SC算法网络结构构建完毕后将进入自主学习阶段, 该阶段采用广义策略迭代思想实现, 策

略评估在未收敛时便与策略改进交替进行,具体步骤如下所示.

Step 1: 目标系统开始一次尝试. 初始化当前状态$S_t = S_0$. 若本次尝试未结束,则循环执行下列步骤:

Step 1.1: 使用式(3)计算S_t在各状态基函数的隶属度向量$\varphi(S_t)$;

Step 1.2: 使用式(2)计算S_t的归一化状态基函数隶属度向量$\Phi(S_t)$;

Step 1.3: 使用式(8)进行策略评估,使用式(7)选择当前状态下应采取的动作A_t,获取即时奖励r,且环境转移到下一状态S_{t+1};

Step 1.4: 使用式(6)计算TD误差Δ;

Step 1.5: 依据式(5)调整Q值函数逼近器参数向量进行策略改进,依据Q值函数逼近器自适应调整过程更新构建样本集和状态基函数. 设置当前状态$S_t = S_{t+1}$.

Step 2: 若本次尝试成功则学习算法结束,否则执行Step 1再次进行尝试.

从以上步骤可以看出,NPAGPI-SC在学习阶段的计算复杂度主要由策略评估复杂度$O(Km)$、策略选择复杂度$O(K)$、策略改进复杂度$O(K + Km + K^2m)$和基函数调整复杂度$O(m)$组成,即$O(2K + (K^2 + 2K + 1)m)$,远小于其他近似策略迭代增强学习算法$O(m^3)$的时间复杂度,可以更好地满足在线计算的要求.

5 一级倒立摆平衡控制仿真实验

5.1 实验描述

一级倒立摆平衡控制是增强学习研究的标准测试问题,本文通过对其进行仿真实验来验证NPAGPI-SC算法的有效性和鲁棒性. 为了方便与已有算法进行对比,采用文献[1]的方法建立目标系统模型

$$\ddot{\theta} = \frac{g \cdot \sin\theta - \dfrac{\cos\theta \cdot (F + \mathrm{MP} \cdot \mathrm{len} \cdot \dot{\theta}^2 \cdot \sin\theta)}{\mathrm{MC} + \mathrm{MP}}}{\mathrm{len} \cdot \left(\dfrac{4}{3} - \dfrac{\mathrm{MC} \cdot \cos^2\theta}{\mathrm{MC} + \mathrm{MP}}\right)}. \tag{35}$$

其中: 系统的状态$S = [\theta \quad \dot{\theta}]^{\mathrm{T}}$,$\theta$为摆杆偏离垂直方向的角度,$\dot{\theta}$为角速度;$F$为作用于小车质心上的力,最大值为50 N,向右取+,向左取−;MC为小车质量,MC = 8.0 kg;MP为摆杆质量,MP = 2.0 kg;len为摆杆长度的一半,len = 0.5 m;g为重力加速度,$g = 9.8\ \mathrm{m/s^2}$.

实验中,仿真时间步设置为0.1 s,折扣率$\gamma = 0.95$. 若倒立摆在一次尝试中保持指定时间步不倒,则认为本次实验成功;否则,若摆杆偏离垂直方向角度超过90°,则认为本次实验失败,相应的即时奖励r如下所示:

$$r = \begin{cases} 0, & |\theta| \leqslant 90°; \\ -1, & \text{otherwise}. \end{cases} \tag{36}$$

5.2 实验结果

5.2.1 NPAGPI-SC取不同参数的实验结果

设置实验目标为倒立摆运行3 000时间步不倒,实验最大尝试次数为300,当$\varepsilon = 0.1$时,离散动作集合分别取2个离散值{−50 N, +50 N}、3个离散值{−50 N, 0 N, +50 N}、4个离散值{−50 N, −25 N, +25 N, +50 N}、5个离散值{−50 N, −25 N, 0 N, +25 N, +50 N}、6个离散值{−50 N, −33 N, −16 N, +16 N, +33 N, +50 N}、7个离散值{−50 N, −33 N, −16 N, 0 N, +16 N, +33 N, +50 N}、8个离散值{−50 N, −37.5 N, −25 N, −12.5 N, +12.5 N, +25 N, +37.5 N, +50 N}、9个离散值{−50 N, −37.5 N, −25 N, −12.5 N, 0 N, +12.5 N, +25 N, +37.5 N, +50 N}、10个离散值{−50 N, −40 N, −30 N, −20 N, −10 N, +10 N, +20 N, +30 N, +40 N, +50 N}、11个离散值{−50 N, −40 N, −30 N, −20 N, −10 N, 0 N, +10 N, +20 N, +30 N, +40 N, +50 N},在以上离散动作集合取值下分别进行100次独立仿真运算,实验结果如表1所示.

表1 离散动作集合在不同取值下的实验结果

离散动作集合	成功率/%	最小尝试次数	最大尝试次数	平均尝试次数	平均样本数量	平均基函数数量(学习前/后)
2个离散值	100	3	76	7.65	167.69	11.92/12.07
3个离散值	100	3	81	11.92	214.61	10.85/11.01
4个离散值	100	3	77	11.61	258.96	11.62/11.75
5个离散值	100	3	85	10.14	256.24	10.74/10.99
6个离散值	100	3	141	9.97	279.65	11.36/11.55
7个离散值	100	3	117	8.25	272.74	10.96/11.19
8个离散值	100	3	131	9.62	271.37	11.26/11.42
9个离散值	100	3	218	11.97	306.39	11.28/11.55
10个离散值	100	3	156	10.95	284.52	11.67/11.91
11个离散值	100	3	60	5.96	285.51	11.16/11.32

由表1可见,不论离散动作集合如何取值,NPAGPI-SC算法均能在较短尝试次数内学习到一级倒立摆系统的平衡控制策略,验证了算法在离散动作集合不同取值下的有效性. 若每个被选择执行的离散动作都叠加[−10 N, +10 N]的均匀噪声,则在离散动作集合的每种取值下分别进行100次独立仿真运算,实验结果如表2所示.

对比表1和表2可以看出,当离散动作叠加噪声后,离散动作集合在不同取值下的采样样本数量和基函数数量均略有增加,但并未影响算法的成功率,且平均尝试次数并未明显增加,显示出算法在离散动作集合不同取值下均具有较好的鲁棒性.

表 2　离散动作集合在不同取值下叠加噪声后的实验结果

离散动作集合	成功率/%	最小尝试次数	最大尝试次数	平均尝试次数	平均样本数量	平均基函数数量(学习前/后)
2个离散值	100	3	63	17.82	300.92	13.1/13.5
3个离散值	100	3	69	11.42	290.21	13.29/13.69
4个离散值	100	3	72	13.4	310.07	13.02/13.37
5个离散值	100	3	83	9.43	307.1	12.79/13.08
6个离散值	100	3	90	12.31	313.09	12.47/12.94
7个离散值	100	3	133	10.4	336.27	12.96/13.33
8个离散值	100	3	92	10.06	294.18	12.39/12.81
9个离散值	100	3	47	8.18	330.02	12.6/12.84
10个离散值	100	3	77	8.98	307.39	12.05/12.34
11个离散值	100	3	74	10.05	326.6	12.39/12.71

设置实验目标为倒立摆运行3 000时间步不倒,实验最大尝试次数为300,当离散动作集合取3个离散值{−50 N, 0 N, +50 N},ε取0.5、0.1、0.05、0.01分别进行100次独立仿真运算,实验结果如表3所示.

表 3　ε在不同取值下的实验结果

ε取值	成功率/%	最小尝试次数	最大尝试次数	平均尝试次数	平均样本数量	平均基函灵敏数量(学习前/后)
0.5	100	3	42	6.18	68.88	8.88/9.09
0.1	100	3	81	11.92	214.61	10.85/11.01
0.05	100	3	101	8.62	396.99	12.2/12.33
0.01	100	3	193	14.19	1715.49	16.95/16.98

由表3可见,随着ε精度的提高,采样样本数量和基函数数量均有所增加,但不论ε如何取值,NPAGPI-SC算法均能在较短尝试次数内学习到一级倒立摆系统的平衡控制策略,验证了算法在ε不同取值下的有效性. 若每个被选择执行的离散动作都叠加[−10 N, +10 N]的均匀噪声,则在不同ε取值下分别进行100次独立仿真运算,实验结果如表4所示.

表 4　动作叠加噪声后ε在不同取值下的实验结果

ε取值	成功率/%	最小尝试次数	最大尝试次数	平均尝试次数	平均样本数量	平均基函灵敏数量(学习前/后)
0.5	100	3	52	9.51	90.46	10.28/11.21
0.1	100	3	69	11.42	290.21	13.29/13.69
0.05	100	3	96	16.31	591.76	15.02/15.37
0.01	100	3	192	17.76	2 880.54	19.51/19.54

对比表3和表4可以看出,当在离散动作上叠加噪声后,ε在不同取值下的采样样本数量和基函数数量均有所增加,但并未影响算法的成功率,且平均尝试次数并未有明显增加,显示出算法在ε不同取值下均具有较好的鲁棒性.

设置实验目标为倒立摆运行3 000时间步不倒,当离散动作集合取3个离散值{−50 N,0 N,+50 N},ε取0.1时进行100次独立仿真运算,学习成功的尝试次数分布如图3所示,典型摆杆角度和角速度的学习过程曲线如图4所示.

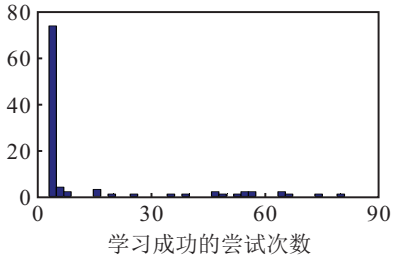

图 3　进行100次仿真运算尝试次数的分布

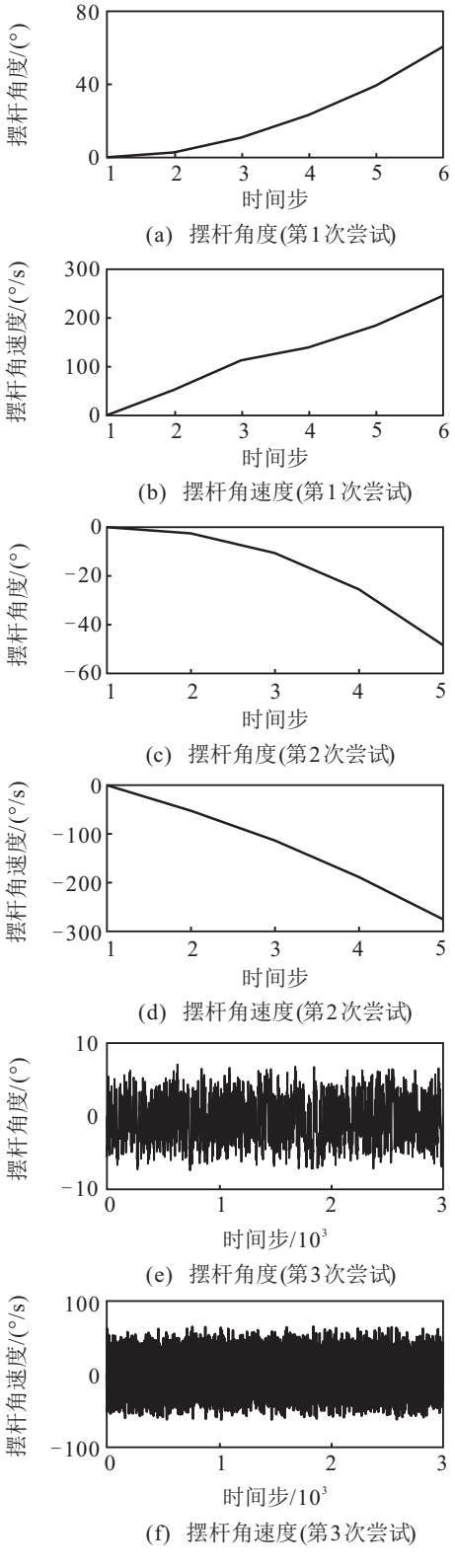

图 4　典型学习过程曲线

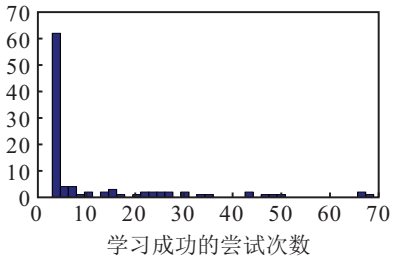

图 5 100次独立仿真运算尝试次数的分布(有噪声下)

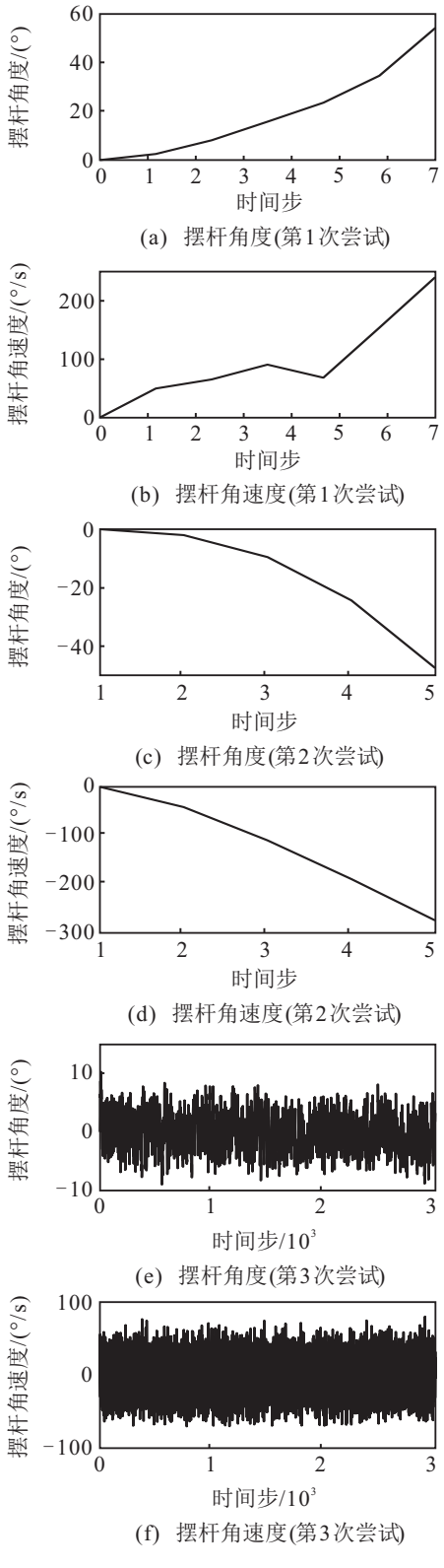

图 6 典型学习过程曲线(有噪声下)

由图3可见,算法基本可在第3次尝试时得到正确的控制策略. 由图4可见,学习成功后摆杆的摆动角度基本稳定在$[-8°,+8°]$,角速度基本稳定在$[-60°/s,+60°/s]$. 以上数据表明算法的学习速度和控制精度都较为理想.

若每个被选择执行的离散动作都叠加[−10 N, +10 N]的均匀噪声,则进行100次独立仿真运算,学习成功的尝试次数分布如图5所示,典型摆杆角度和角速度的学习过程曲线如图6所示.

由图5可见,算法基本可在第3次尝试时得到正确的控制策略. 由图6可见,学习成功后摆杆的摆动角度基本稳定在$[-10°,+10°]$,摆杆角速度基本稳定在$[-80°/s,+80°/s]$. 通过对离散动作叠加噪声前后的典型学习过程曲线进行比较可以看出,噪声对NPAGPI-SC算法学习速度和控制精度未造成较大影响,显示出算法在控制性能方面具有较强的鲁棒性.

5.2.2 与典型近似策略迭代增强学习算法的对比

本节依旧采用一级倒立摆平衡控制仿真实验比较各类近似策略迭代增强学习算法的性能. 为了与文献[3]和文献[6]的实验条件保持一致,使离散动作集合取3个离散值{−50 N,0 N,+50 N},且每个被选择执行的离散动作都叠加[−10 N,+10 N]的均匀噪声.

online LSPI、BLSPI等在线近似策略迭代增强学习算法一般通过不同参数条件下获得控制策略的速度进行评价. 设置实验目标为倒立摆运行3 000时间步不倒,将ε分别为0.5、0.1、0.05、0.01时,NPAGPI-SC进行100次独立仿真运算的结果与onlineLSPI、BLSPI算法的最优实验结果[3]进行比较,3种算法在不同尝试次数下的平均平衡时间步数如表5所示.

表 5 与典型在线近似策略迭代增强学习算法的对比

算 法	50次尝试	100次尝试	150次尝试	200次尝试	250次尝试	300次尝试
NPAGPI-SC $\varepsilon=0.5$	2 980	3 000	3 000	3 000	3 000	3 000
NPAGPI-SC $\varepsilon=0.1$	2 944	3 000	3 000	3 000	3 000	3 000
NPAGPI-SC $\varepsilon=0.05$	2 841	3 000	3 000	3 000	3 000	3 000
NPAGPI-SC $\varepsilon=0.01$	2 775	2 820	2 955	3 000	3 000	3 000
online LSPI	27	35	36	37	928	2 112
BLSPI LSTD-Q	921	1 800	2 187	2 576	2 771	3 000
BLSPI LSPE-Q	34	1 532	2 701	3 000	3 000	3 000

由表5可见,随着ε精度的提高,NPAGPI-SC算法的学习速度逐渐降低,但总体性能要优于online LSPI和BLSPI.

LSPI与KLSPI等离线近似策略迭代增强学习算法一般通过在不同样本数目条件下策略迭代获得控制策略的性能进行评价[6]. 从实现方法上看,LSPI和KLSPI在每次迭代时都将样本数据累加到矩阵中,其

一次迭代过程所需的样本数量大致与NPAGPI-SC一次尝试相当,因此NPAGPI-SC的平均尝试次数基本等价于LSPI与KLSPI的平均迭代次数. 当$\varepsilon = 0.1$时,NPAGPI-SC独立运行100次的平均采样数量为290个,与文献[6]中50周期(约300个样本)的样本数量相当. 选择与文献[6]相同的实验方法,即策略性能评价的最大运行实际步长为2 000、每个策略独立运行1 000次,得到的实验结果对比如表6所示.

表6　采用50周期训练样本获得的策略性能

算　法	平均迭代次数	最小时间步	最大时间步	平均时间步
NPAGPI-SC $\varepsilon = 0.1$	9.32	2 000	2 000	2 000
LSPI	20	6	2 000	460
KLSPI	20	898	2 000	1 997.9

表6中NPAGPI-SC的平均时间步由100次独立仿真结果中随机抽取10个所学策略分别运行1 000次得到的均值计算. 当$\varepsilon = 0.05$时,NPAGPI-SC独立运行100次的平均采样数量为592个,与文献[6]中100周期(约600个样本)的样本数量相当. 选择与文献[6]相同的实验方法,得到的实验结果对比如表7所示.

表7　采用100周期的训练样本获得的策略性能

算　法	平均迭代次数	最小时间步	最大时间步	平均时间步
NPAGPI-SC $\varepsilon = 0.05$	11.95	2 000	2 000	2 000
LSPI	20	6	2 000	896
KLSPI	20	2 000	2 000	2 000

实验数据表明,无论采用50周期的训练样本还是100周期的训练样本,NPAGPI-SC算法都可以在更少的平均迭代次数下达到或超过LSPI和KLSPI算法的策略性能.

6　结　论

本文实现了一种基于状态聚类的非参数化近似广义策略迭代增强学习算法NPAGPI-SC. 相较典型的近似策略迭代增强学习算法,NPAGPI-SC不仅计算复杂度小,且只需指定增强学习的允许误差率便能完全自主地构建、调整基函数和参数,学习到目标系统的控制策略. 一级倒立摆平衡控制的仿真实验结果验证了所提出的算法在不同参数下的有效性和鲁棒性,相较于目前典型的近似策略迭代增强学习算法,NPAGPI-SC具有收敛速度更快的优势.

参考文献(References)

[1] Lagoudakis Michail G, Parr Ronald. Least squares policy iteration[J]. J of Machine Learning Research, 2003, 4(6): 1107-1149.

[2] Busoniu Lucian, Ernst Damien, Schutter Bart De, et al. Online least-squares policy iteration for reinforcement learning control[C]. Proc of the 2010 American Control Conf. Baltimore: IEEE Press, 2010: 486-491.

[3] 周鑫, 刘全, 傅启明, 等. 一种批量最小二乘策略迭代方法[J]. 计算机科学. 2014, 41(9): 232-238.
(Zhou X, Liu Q, Fu Q M, et al. Batch least-squares policy iteration[J]. Computer Science, 2014, 41(9): 232-238.)

[4] Ormoneit Dirk, Sen Saunak. Kernel-based reinforcement learning[J]. Machine Learning, 2002, 49(2): 161-178.

[5] Xu Xin, Hu Dewen, Lu Xicheng. Kernel-based least squares policy iteration for reinforcement learning[J]. IEEE Trans on Neural Networks, 2007, 18(4): 973-992.

[6] Xu Xin, Peng Chengzhang, Dai Bin, et al. A kernel-based reinforcement learning approach to stochastic pole balancing control system[C]. IEEE/ASME Int Conf on Advanced Intelligent Mechatronics. Montreal: IEEE Press, 2010: 1329-1334.

[7] 程玉虎, 冯涣婷, 王雪松. 基于状态-动作图测地高斯基的策略迭代强化学习[J]. 自动化学报. 2011, 37(1): 44-51.
(Cheng Y H, Feng H T, Wang X S. Policy iteration reinforcement learning based on geodesic gaussian basis defined on state-action graph[J]. Acta Automatica Sinica, 2011, 37(1): 44-51.)

[8] Kim Min-soeng, Hong Gun-gi, Lee Ju-jang. On-line fuzzy Q-learning with extended rule interpolation technique[C]. IEEE/RSJ Int Conf on Intelligent Robots and Systems. Kyongju: IEEE Press, 1999, 2: 757-762.

[9] Sun Haojun, Wang Shengrui, Jiang Qingshan. FCM-Based model selection algorithms for determining the number of clusters[J]. Pattern Recognition, 2004, 37(10): 2027-2037.

[10] Kaufman Leonard, Rousseeuw Peter J. Finding groups in data: An introduction to cluster analysis[M]. New York: John Wiley and Sons Ltd, 1990: 68-125.

[11] 于剑, 程乾生. 模糊聚类方法中的最佳聚类数的搜索范围[J]. 中国科学: E辑. 2002, 32(2): 274-280.
(Yu J, Cheng Q S. The upper bound of the optimal number of clusters in fuzzy clustering[J]. Science in China: Series E, 2002, 32(2): 274-280.)

（责任编辑：郑晓蕾）