第 21 卷 第 2 期　　控 制 与 决 策　　2006 年 2 月
Vol. 21 No. 2　　Control and Decision　　Feb. 2006

文章编号: 1001-0920(2006)02-0143-05

连续状态自适应离散化基于 K-均值聚类的强化学习方法

文　锋, 陈宗海, 卓　睿, 周光明
(中国科学技术大学 自动化系, 合肥 230027)

摘　要: 使用聚类算法对连续状态空间进行自适应离散化, 得到了基于 K-均值聚类的强化学习方法. 该方法的学习过程分为两部分: 对连续状态空间进行自适应离散化的状态空间学习, 使用 K-均值聚类算法; 寻找最优策略的策略学习, 使用替代合适迹 Sarsa 学习算法. 对连续状态的强化学习基准问题进行仿真实验, 结果表明该方法能实现对连续状态空间的自适应离散化, 并最终学习到最优策略. 与基于 CMAC 网络的强化学习方法进行比较, 结果表明该方法具有节省存储空间和缩短计算时间的优点.

关键词: 强化学习; K-均值聚类算法; Sarsa 学习; 连续状态; 自适应离散化
中图分类号: TP13　　**文献标识码**: A

Reinforcement Learning Method of Continuous State Adaptively Discretized Based on K-means Clustering

WEN Feng, CHEN Zong-hai, ZHUO Rui, ZHOU Guang-ming
(Department of Automation, University of Science and Technology of China, Hefei 230027, China. Correspondent: CHEN Zong-hai, E-mail: chenzh@ustc.edu.cn)

Abstract: A K-means clustering based reinforcement learning method is proposed, which uses clustering algorithm to adaptively discretize continuous state space. The learning of this method is divided into two processes, state space learning using K-means clustering algorithm for adaptive discretization of continuous states and policy learning using Sarsa algorithm for finding optimal policy. Simulation conducted on reinforcement learning benchmark problem with continuous state shows that the proposed method can adaptively discretize continuous state space and learn optimal policy in the end. Comparison with CMAC network based reinforcement learning method shows that the proposed method has advantages of saving memory and reducing computation time.

Key words: Reinforcement learning; K-means clustering algorithm; Sarsa learning; Continuous state; Adaptively discretized

1 引　　言

实际中遇到的顺序决策问题, 如过程控制、机器人控制等, 其中涉及到的动态系统的状态变量一般取值于连续实数区间, 如温度、流量、位置、速度等. 这些变量的取值可能有无数个, 因而状态总数是无限的. 标准的强化学习方法[1,2]只适用于有限离散状态问题, 要应用于上述领域, 需要解决连续状态的表示问题.

应用强化学习的思想解决连续状态的表示问题主要有两类方法: 基于参数化表示的函数逼近方法和基于离散化的方法. 前者的参数调整过程比较复杂, 甚至可能出现不收敛的情况[1]; 而简单地对连续状态进行离散化, 分割所得状态总数面临"维数灾难"问题. 实际上, 状态空间中与学习目标相关的状态一般只占整个状态空间的极少部分, 大多数状态与问题的解决无关. 因此在整个状态空间进行等精度离散化是不必要的.

自适应离散化方法可根据需要对状态空间进行

收稿日期: 2005-01-10; **修回日期**: 2005-03-05.
基金项目: 国家自然科学基金项目(60575033); 国家高水平大学 985 计划项目(KY2701).
作者简介: 文锋(1978—), 男, 湖南祁东人, 博士生, 从事智能控制等研究; 陈宗海(1963—), 男, 安徽桐城人, 教授, 博士生导师, 从事复杂系统建模、仿真与控制等研究.

可变精度划分, 避免了简单离散化的缺陷, 从而有效地解决了连续状态的表示问题. 称这类强化学习方法为连续状态自适应离散化强化学习方法. 已有的方法包括可变分辨率方法[3]、基于树表示方法[4]、自组织方法[5~9]等, 但这些方法实现起来比较复杂.

本文选用实现简单的聚类算法解决连续状态的表示问题. 文献[10]提出一种基于状态聚类的强化学习方法, 利用先验知识或者事先训练控制器, 得到状态空间的划分后再进行强化学习; 本文方法则同时在线进行聚类和强化学习, 省略了离线的聚类学习过程. 本文将 K-均值聚类算法与 Sarsa 算法相结合, 提出了基于 K-均值聚类的强化学习方法. 该方法能根据输入状态的分布自动调整聚类中心, 实现对连续空间的自适应离散化. 文中对聚类算法参数以及训练方案的选择进行讨论, 并将该方法应用于 MountainCar 问题, 仿真实验结果表明该方法能学习到最优策略. 同基于 CMAC 网络的强化学习方法进行对比实验, 结果表明该方法具有节省存储空间和缩短计算时间的优点.

2 连续状态自适应离散化的强化学习方法

将 K-均值聚类算法用于连续状态空间的自适应离散化, 并与 Sarsa 算法相结合, 可得到一种自适应离散化强化学习方法. 其学习过程分为两部分: 状态空间学习和策略学习.

2.1 K-均值聚类算法

K-均值聚类算法是一种无监督学习方法, 它可将一组数据点剖分成几个不同的部分, 每一部分中的数据都尽量有相同的性质. 这里将 K-均值聚类算法用于状态空间学习, 该算法能根据训练数据的分布调整聚类中心, 将训练数据分成不同类别. 具体算法参见文献[11].

2.2 Sarsa 学习方法[1]

强化学习的目标是使得到的回报之和最大, 即最大化 $g(t)=\sum_{k=1}^{\infty}\gamma^{k}r_{t+k}$. 其中 r_t 为在 t 时刻获得的回报, γ 为折扣系数, $0\le\gamma\le 1$. 引入 cost-to-go 函数 $g(t)$, 表示从 t 时刻起所获得的回报之和. 根据 Bellman 最优原理, 在时刻 t 使动作 a_t 最优, 也使 cost-to-go 函数 $g(t)$ 最大化. cost-to-go 函数有值函数和 Q 函数两种形式. Sarsa 学习算法使用 Q 函数形式, 表示在状态 s_t 下, 采取动作 a_t 后得到的回报总和为

$$Q(s_t,a_t)=r_{t+1}+\gamma V(s_{t+1})=r_{t+1}+\gamma\min_i Q(s_{t+1},a_i). \tag{1}$$

得到 Q 函数的估计后, 由贪婪算法可直接得到最优策略. Sarsa 学习算法参见文献[1].

为了加快学习, Sarsa 学习算法使用了替代合适迹. 对于每个状态动作对 (s,a), 定义合适迹 $e(s,a)$ 更新为

$$e(s,a)=\begin{cases}1, & s=s_t,\ a=a_t;\\ 0, & s=s_t,\ a\neq a_t;\\ \gamma\lambda e(s,a), & \text{otherwise}\end{cases} \tag{2}$$

其中: γ 为前面定义的折扣系数, λ 为合适迹衰减参数, $0<\lambda<1$.

对于任意动作对 (s_k,a_i), Q 函数更新方程变成

$$Q(s_k,a_i)\leftarrow Q(s_k,a_i)+\beta e(s_k,a_i)\times[r+\gamma Q(s',a')-Q(s,a)]. \tag{3}$$

上述 Sarsa 学习算法只适用于有限离散状态空间和动作空间问题, 而不能直接应用于连续状态空间问题.

2.3 基于 K-均值聚类的强化学习方法

将 K-均值聚类算法与替代合适迹 Sarsa 学习算法相结合, 可得到基于 K-均值聚类的强化学习方法. 使用聚类算法实现对连续状态空间自适应离散化的过程称为状态空间学习; 根据回报信号学习最优策略的过程称为策略学习.

定义 1 设 $x\in R^n$ 为对象的状态, 其中 n 为状态维数, 则称 x 为实际状态.

实际状态在状态空间中连续分布, 不能直接应用标准的强化学习方法. 可使用 K-均值聚类算法离散化状态空间, 解决实际连续状态的表示问题.

定义 2 定义聚类中心 t_j 所对应的状态空间单元 $R_j\subset R^n$ 为

$$R_j=\{x\in R^n\mid \arg\min_k\|x-t_k\|=j,\ k=1,2,\cdots,m\}.$$

定理 1 给定 m 个聚类中心的集合 $T=\{t_k,k=1,2,\cdots,m\}$, 可得到对状态空间的一个划分, 记为 $P=\{R_j\mid R_j\subset R^n,j=1,2,\cdots,m\}$. 其中 R_j 为聚类中心 t_j 所对应的状态空间单元, 满足

$$\bigcup_{j=1}^{m}R_j=R^n,\ R_i\cap R_j=\Phi,\quad \forall i\neq j,\ i,j=1,2,\cdots,m.$$

证明由定义 2 易得.

定义 3 给定状态空间的划分 P, 定义离散化函数 $s(\cdot):R^n\to S$ 为

$$s(x)=s_j,\ x\in R_j.$$

其中: s_j 称为离散状态, 简称状态; 集合 $S=\{s_j,j=1,2,\cdots,m\}$ 称为离散状态集.

在状态空间学习中, K-均值聚类算法能按实际状态的分布调整聚类中心的位置, 最终实现对状态空间的自适应离散化. 经过离散化后, 可用图 1 所示

的表格形式 Q-函数, 假设动作空间为有限离散集合. 在此基础上, 使用替代合适迹 Sarsa 学习算法进行策略学习.

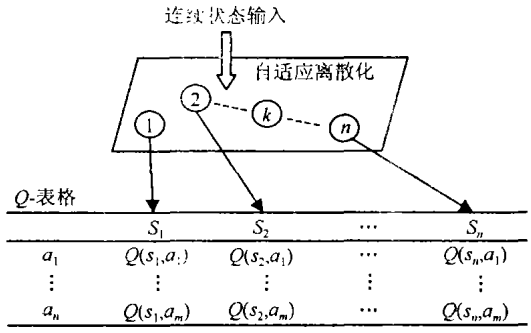

图 1　离散化后的 Q-函数表示

将状态空间学习与策略学习相结合, 可得到一种在线学习算法如下:

1) 初始化所有状态和动作的 Q-函数值以及聚类中心 t_k.

2) 初始化对象的状态 x.

3) 对状态 x 确定最匹配单元, 设编号为 $j(x)$, 对应于状态 s_j, 即

$$j(x) = \arg\min_k \| x(n) - t_k(n) \|, \quad k = 1, 2, \cdots, m. \tag{4}$$

其中: $t_k(n)$ 为第 n 次迭代中第 k 个聚类中心的位置, 范数定义为 Euclidean 距离.

4) 在聚类算法中, 根据输入 x 调整聚类中心

$$t_k(n+1) = \begin{cases} t_k(n) + \eta[x(n) - t_k(n)], \\ \qquad k = j(x); \\ t_k(n), \text{otherwise} \end{cases} \tag{5}$$

其中 η 为聚类算法的学习率, $0 < \eta < 1$.

5) 采用 ϵ-贪婪策略选取动作

$$a = \begin{cases} \arg\max_i Q(s_j, a_i), \text{with prob } 1-\epsilon, \\ \text{random action, with prob } \epsilon \end{cases} \tag{6}$$

6) 使用式(2) 更新合适迹.

7) 应用动作 a, 观察所得的回报 r 和下一个状态 x'.

8) 对状态 x' 确定最匹配单元, 设编号为 $j'(x')$, 对应于状态 $s_{j'}$.

9) 在聚类算法中, 根据输入 x' 并用式(5) 调整聚类中心 t_k.

10) 采用 ϵ-贪婪策略(6) 选择动作 a'.

11) 对所有离散状态 s_k 和动作 a_i, 更新 Q-函数

$$Q(s_k, a_i) \leftarrow Q(s_k, a_i) + \beta e(s_k, a_i) \times [r + \gamma Q(s_{j'}, a') - Q(s_j, a)]. \tag{7}$$

12) $x \leftarrow x'$, $a \leftarrow a'$.

13) 若 x 为非中止状态, 则返回 6).

3　MountainCar 问题[12]

以强化学习的一个基准问题 ——MountainCar 问题作为实验对象. 如图 2 所示, 车辆在一山区道路上行驶, 目标是冲上右边的山顶. 由于重力的影响大于引擎的驱动能力, 即使开到最大动力也不能直接冲上山顶. 唯一可行的方法是先倒退, 然后向前冲, 以积累足够的能量爬上山坡. 这个问题说明要达到目标(冲上山顶), 首先要使情况变坏(倒退行驶, 远离目标), 然后才能变得更好. 一般的控制方法除非加入特殊设计, 否则很难处理这类问题.

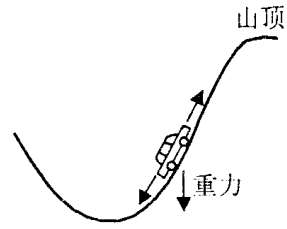

图 2　MountainCar 问题

该问题有两个连续分布的状态分量, 即 $x_t = [p_t, v_t]^T$, 其中 p 为小车位置, v 为小车速度. 动作 $a \in \{-1, 0, +1\}$, 分别对应于向后加速、不加速和向前加速. 将山坡的地形简化为 $h = \sin(3p_t)$, 其中 h 为高度. 得到小车简化的运动模型

$$\begin{cases} v_{t+1} = \text{bound}[v_t + 0.01a_t + g\cos(3p_t)], \\ p_{t+1} = \text{bound}[p_t + v_t]. \end{cases} \tag{8}$$

其中: bound 表示范围限制, $p \in [-1.2, 0.5]$, $v \in [-0.07, 0.07]$, $g = -0.0025$. 当位置 p_{t+1} 超出范围时, 将速度 v_{t+1} 设置为 0. 位置 $p = -0.5$ 对应于山谷的最低点, 小车需要从这一点以初速度 0 出发, 即位置 $p > 0.5$.

设计回报信号

$$r_t = \begin{cases} -1, p_t \le 0.5; \\ 0, p_t > 0.5. \end{cases} \tag{9}$$

强化学习将使所得的回报之和最大, 即小车爬上山顶所花费的时间最短.

4　仿真实验

基于 K-均值聚类的强化学习方法有状态空间学习和策略学习两个过程, 各参数的选择非常复杂. 为讨论聚类算法参数的影响, 选择 Singh 在解决 MountainCar 问题时所采用的参数值[12]: $\epsilon = 0$, $\gamma = 1.0$, $\lambda = 0.9$, $\beta = 0.5$, 并将基于 CMAC 网络的强化学习方法[12] 与本文的强化学习方法进行比较.

4.1　聚类中心初始分布的选择

状态空间学习与策略学习相互影响, 其中聚类算法的训练数据由策略学习得到的策略在线产生, 其分布并不固定. 聚类中心在整个空间中随机分布,

聚类中心之间的距离可能过大, 离散状态对应的区域也过大, 因此可能陷入循环状态. 即小车在山谷中来回运动, 永远不能到达山顶. 现解释如下:

设从初始实际状态出发, 不管选择哪种动作, 实际状态转移后, 仍处于同一离散状态 s_0. 设状态转移前后, 选择的动作分别为 a_p 和 a_q, 根据 Sarsa 学习算法(令 $\gamma = 1.0$), 有

$$Q(s_0, a_p) \leftarrow Q(s_0, a_p) + \beta[-1 + Q(s_0, a_q) - Q(s_0, a_p)] \quad (10)$$

1) 若 $a_p = a_q$, 则更新式(10) 中的 $-1 + Q(s_0, a_q) - Q(s_0, a_p) = -1, Q(s_0, a_p)$ 将减小 β. 这样持续下去, 在某个时刻后, 动作 a_p 将不再是状态 s_0 下的贪婪动作, 而出现情况 2).

2) $a_p \neq a_q$, 由 1) 中的分析知, 状态 s_0 下的原最大 Q 值 $Q(s_0, a_p)$ 减小 β 后, 动作 a_q 将成为贪婪动作, 即 $Q(s_0, a_q)$ 将成为状态 s_0 下的最大 Q 值. 显然有 $|Q(s_0, a_q) - Q(s_0, a_p)| < \beta < 1$, 故 $-1 + Q(s_0, a_q) - Q(s_0, a_p) < 0$, 因此 $Q(s_0, a_q)$ 也将减小. 在下一时刻, a_q 可能变成非贪婪动作.

于是, 在离散状态 s_0 下, 若交替选择各动作, 小车就不能积累足够的能量冲上山顶, 而在山谷中来回运动. 可见, 聚类中心的初始分布要集中在初始状态附近. 实验开始时, 实际状态轨迹从初始值出发, 聚类算法随状态的分布自动调整聚类中心, 从而避免这种现象.

4.2 聚类中心数对学习的影响

聚类中心数决定了状态空间离散化的划分总数. 虽然 K-均值聚类算法能根据数据的分布自动调整聚类中心的分布, 但总聚类中心数会影响聚类算法所能进行调整的能力. 聚类中心数分别取 30, 50 和 80, 在不同的学习率下进行实验, 结果如图 3 所示.

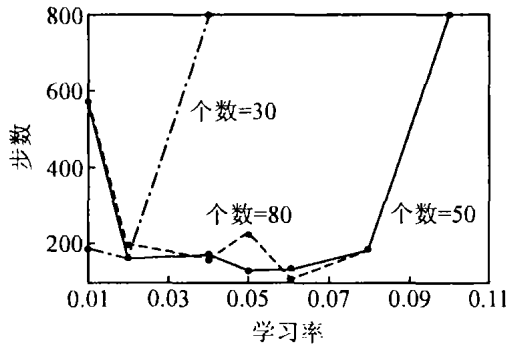

图 3 不同聚类中心数在不同学习率下的学习效果

可见, 聚类中心较多, 达到较好学习效果的学习率范围较大. 聚类中心个数过少, 聚类算法对状态空间的划分比较粗略, 甚至可能陷入循环状态. 但聚类中心越多, 算法的计算时间越长, 并且学习效果并不一定更好. 因此需要尝试使用不同的聚类中心个数, 选择最优值.

4.3 与基于 CMAC 网络的强化学习方法对比

在解决 MountainCar 问题中, 基于 CMAC 网络的强化学习方法[12] 的策略学习部分使用替代合适迹 Sarsa 学习算法, 并且参数相同. CMAC 网络在各状态分量上分为 8 等分, 层数为 5. 本文采用基于 K-均值聚类的强化学习方法, 聚类中心数为 50, 学习率为 0.05.

在相同的初始状态下, 两种方法的学习效果如图 4 所示. 基于 K-均值聚类的强化学习方法, 虽然在初始阶段表现较差, 但是最终的学习结果相同, 并且结果更加稳定.

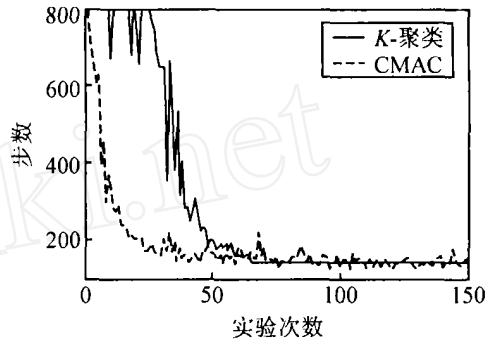

图 4 本文方法与基于 CMAC 方法的学习效果比较

本文提出的方法优于基于 CMAC 网络的方法, 具体体现在: 1) 在 Sarsa 学习算法中, 需要存储 Q 值和合适迹值, 存储单元个数分别与聚类中心个数和 CMAC 网络单元个数相同. 其中前者为 50, 后者为 320, 后者是前者的 6.4 倍. 可见本文方法大大减少了存储空间. 2) 进行 5 组实验, 使用 Matlab 中的 cputime 函数计算算法的运行时间, 实验数据见表 1. 相比而言, 本文方法的平均运行时间缩短了 37%.

表 1 本文方法与基于 CMAC 网络的方法运行时间比较

算 法	实验 1	实验 2	实验 3	实验 4	实验 5	平均值	方差
CMAC 网络	25.87	23.19	19.51	21.39	21.54	22.30	2.38
K-均值聚类	14.55	12.40	14.49	15.58	12.45	13.89	1.41

基于 K-均值聚类的强化学习方法所学习到的最优状态轨迹(曲线) 和聚类中心(点) 分布如图 5 所示. 其中聚类中心主要分布在最优轨迹附近, 在其他区域分布则很少. 图中同时显示出各聚类中心所对应的区域单元, 其中初始状态点附近的离散化比较精细, 而靠外区域的离散化比较粗略. 这与上节的分析结果相符合, 表明本文方法实现了对状态空间的自适应离散化.

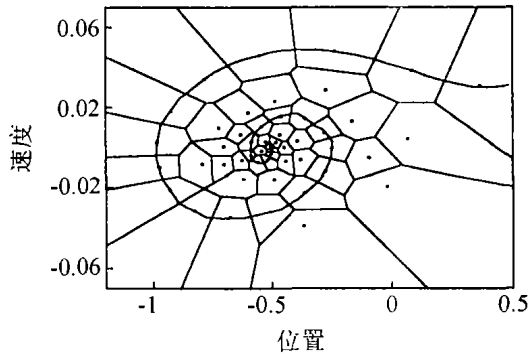

图 5　基于 K-均值聚类的强化学习方法的学习结果

5　结　　论

本文使用 K-均值聚类算法实现连续状态空间的自适应离散化. 与替代合适迹 Sarsa 学习算法相结合, 得到基于 K-均值聚类的强化学习方法. 该方法包含状态空间学习和策略学习两个过程. 前者决定了后者用于决策的 cost-to-go 函数估计的准确程度; 后者则影响前者训练数据的分布. 因此在选择参数时, 要通过尝试选择最佳参数. 将其应用于解决连续状态的 MountainCar 问题, 仿真实验表明, 该方法能实现状态空间的自适应离散化, 并学习到最优策略. 与基于 CMAC 网络的强化学习方法[12]进行比较, 学习效果相当, 但对存储空间需求更少, 计算时间更短, 实现更方便.

参考文献(References)

[1] Sutton R S, Barto A G. *Reinforcement Learning: An Introduction* [M]. Cambridge: MIT Press, 1998.

[2] 高阳, 陈世福, 陆鑫. 强化学习研究综述[J]. *自动化学报*, 2004, 30(1): 86-100.
(Gao Y, Chen S F, Lu X. Research on Reinforcement Learning Technology: A Review [J]. *Acta Automatica Sinica*, 2004, 30(1): 86-100.)

[3] Moore A W, Atkeson C G. The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-spaces [J]. *Machine Learning*, 1995, 21(3): 199-233.

[4] Uther W T, Veloso M M. Tree Based Discretization for Continuous State Space Reinforcement Learning [A]. *AAAI'98*[C]. Madison, 1998: 769-774.

[5] Smith A J. Applications of the Self-organising Map to Reinforcement Learning[J]. *Neural Networks*, 2002, 15(8-9): 1107-1124.

[6] 晏雄伟, 邓志东, 孙增圻. 竞争式 Takagi-Sugeno 模糊再励学习[J]. *自动化学报*, 2002, 28(6): 873-880.
(Yan X W, Deng Z D, Sun Z Q. Competitive Takagi-Sugeno Fuzzy Reinforcement Learning [J]. *Acta Automatica Sinica*, 2002, 28(6): 873-880.)

[7] 路兆梅, 匡文生. 自组织增强学习模糊神经网络控制器的设计[J]. *东南大学学报*, 1999, 29(4): 109-112.
(Lu Z M, Kuang W S. Design of Fuzzy Neural Network Controller Used Reinforcement Learning[J]. *J of Southeast University*, 1999, 29(4): 109-112.)

[8] 马勇, 杨煜普, 许晓鸣, 等. 一类再励学习控制器设计及其在倒车模型中的应用[J]. *上海交通大学学报*, 2000, 34(12): 1661-1663.
(Ma Y, Yang Y P, Xu X M, et al. Design of the Controller Based on Reinforcement Learning and Its Application on Truck Backer-upper [J]. *J of Shanghai Jiaotong University*, 2000, 34(12): 1661-1663.)

[9] Lee I S K, Lau H Y K. Adaptive State Space Partitioning for Reinforcement Learning [J]. *Engineering Applications of Artificial Intelligence*, 2004, 17(6): 577-588.

[10] 李春贵, 吴沧浦, 刘永信. 一种基于状态聚类的 SARSA(λ)强化学习算法[J]. *计算机工程*, 2003, 29(5): 37-98.
(Li C G, Wu C P, Liu Y X. SARSA(λ) Algorithm of Reinforcement Learning Based on States Clustering [J]. *Computer Engineering*, 2003, 29(5): 37-98.)

[11] Haykin S. *Neural Networks: A Comprehensive Foundation* [M]. Beijing: Tsinghua University Press, 2001.

[12] Singh S P, Sutton R S. Reinforcement Learning with Replacing Eligibility Traces [J]. *Machine Learning*, 1996, 22(2): 123-158.

(上接第 142 页)

(Chai T Y. Multivariable Indirect Adaptive Decoupling Control Algorithm [J]. *Acta Automatica Sinica*, 1981, 17(2): 182-190.)

[8] Cabrera J B D, Narendra K S. Issues in the Application of Neural Networks for Tracking Based on Inverse Control [J]. *IEEE Trans on Automatic Control*, 1999, 44(11): 2007-2027.

[9] Chen L, Narendra K S. Identification and Control of a Nonlinear Dynamical System Based on Its Linearization: Part Ⅱ[A]. *Proc of the American Control Conf* [C]. Florida, 2002: 382-387.

[10] 柴天佑. *多变量自适应解耦控制及应用*[M]. 北京: 科学出版社, 2001.
(Chai T Y. *Multivariable Adaptive Decoupling Control and Its Applications*[M]. Beijing: Science Press, 2001.)