一种基于随机投影的贝叶斯时间差分算法

刘 全[1,2,3],于 俊[1,3],王 辉[1,3],傅启明[1,3],朱 斐[1,3]
(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;
3. 软件新技术与产业化协同创新中心,江苏南京 210023)

摘 要: 在强化学习方法中,大部分的算法都是基于值函数评估的算法. 高斯过程时间差分算法利用贝叶斯方法来评估值函数,通过贝尔曼公式和贝叶斯规则,建立立即奖赏与值函数之间的概率生成模型. 在状态空间中,通过在线核稀疏化并利用最小二乘方法来求解新样本的近似线性逼近,以提高算法的执行速度,但时间复杂度依然较高. 针对在状态空间中近似状态的选择问题,在高斯过程框架下提出一种基于随机投影的贝叶斯时间差分算法,该算法利用哈希函数把字典状态集合中的元素映射成哈希值,根据哈希值进行分组,进而减少状态之间的比较. 实验结果表明,该方法不仅能够提高算法的执行速度,而且较好地平衡了评估状态值函数精度和算法执行时间.

关键词: 强化学习; 马尔科夫决策过程; 高斯过程; 随机投影; 时间差分算法
中图分类号: TP181 **文献标识码:** A **文章编号:** 0372-2112 (2016)11-2752-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.11.026

A Bayesian Temporal Difference Algorithm Based on Random Projection

LIU Quan[1,2,3], YU Jun[1,3], WANG Hui[1,3], FU Qi-ming[1,3], ZHU Fei[1,3]
(1. *School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu* 215006, *China*;
2. *Key Laboratory of Symbolic Computation and Knowledge Engineering of Jilin University, Ministry of Education, Jilin University, Changchun, Jilin* 130012, *China*;
3. *Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, Jiangsu* 210023, *China*)

Abstract: Most algorithms are based on policy evaluation in reinforcement learning. The Gaussian process temporal difference is an algorithm that uses Bayesian solution to evaluate value functions. In the method, Gaussian process builds a probabilistic generative model between the immediate reward and the value function through Bellman Equation and Bayesian rule. In order to improve the efficiency of the algorithm, approximate linear approximation for new samples is solved by online kernel sparse and least squares in state space. However, the time complexity is still high. To deal with this problem, a Bayesian temporal difference algorithm bases on random projection algorithm is proposed. The elements in dictionary state set are mapped to hash values by hash function. According to the hash values, groups are divided and the comparison between the states is reduced. The experimental results show that this algorithm not only improves the execution speed, but also obtains balance between execution time and precision of the state value function.

Key words: reinforcement learning; markov decision process; gaussian process; random projection; temporal difference learning

1 引言

强化学习(Reinforcement Learning, RL)是在未知、动态环境中在线求解最优策略,以获取最大期望回报的一类算法. 强化学习方法的基本框架为:Agent 通过试错与环境进行交互,将每一步的延迟回报通过时间信用分配机制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最优策略[1,2]. 目前强化学习方法越来越多地被用于在线控制、作业调度、游戏等领域[3,4].

收稿日期:2015-04-08;修回日期:2015-08-17;责任编辑:蓝红杰
基金项目:国家自然科学基金(No. 61272005, No. 61303108, No. 61373094, No. 61472262, No. 61502323, No. 61502329);江苏省自然科学基金(No. BK2012616);江苏省高校自然科学研究项目(No. 13KJB520020);吉林大学符号计算与知识工程教育部重点实验室项目(No. 93K172014K04);苏州市应用基础研究计划工业部分(No. SYG201422, No. SY201308)

马尔科夫决策过程(Markov Decision Process,MDP)是一类重要的随机过程,经常用来对强化学习进行建模[5]. Sutton 在 1998 年提出对马尔科夫链学习的理论和 TD(λ)算法[6]. 核方法在监督学习和无监督学习问题中都得到了广泛的研究[7]. 目前基于核的强化学习理论与应用成果还较少,这主要是由于核方法需要随机或重复的获取训练样本[8]. 直到 2002 年,Ormoneit 等人提出了基于核的强化学习方法[9]. 后来,Xu 等人提出了基于核的最小二乘 TD 方法(Kernel-based Least Squares TD,KLSTD),将基于核的逼近与 LSTD 相结合[10],取得了一定的效果. 在 KLSTD 基础之上,Xu 等人继续提出了 KLSPI 及 KLSPI-Q 算法[11],并证明了方法的有效性. Yaakov Engel 等人提出了一种新的值函数评估方法,该方法利用核方法来估计值函数,选择核方法中的高斯过程(Gaussian process)模型[12]为值函数建模,通过高斯过程与时间差分方法相结合得到高斯过程的时间差分(Gaussian Process Temporal Difference,GPTD)学习算法[13,14],建立值函数的概率生成模型,然后根据先验,以及观测到的样本,利用贝叶斯推理得到值函数完整的后验分布.

对于固定的策略,GPTD 能够较准确的评估该策略的值函数,但是 GPTD 算法的明显缺点是模型的学习完全依赖于样本,计算量较大. Engel 等人提出了依赖于特征空间的在线核稀疏化方法,将核函数看作是在高维希伯尔特空间上的两个向量的内积,直接去除那些能够用特征空间中特征近似线性逼近的样本[15],利用最小二乘方法来求解新样本的近似线性逼近,以提高时间和空间效率.

本文针对在强化学习状态空间中需要选择近似状态的问题,在高斯过程框架上提出一种基于随机投影的贝叶斯时间差分算法(Bayesian Temporal Difference algorithm based on Random Projection,RPGPTD). 该算法对于新状态,首先进行预处理,把状态转变为二进制编码,使得相似的数据对象,其二进制编码也相似,在此基础上进行相似性比较选择,同时设置参数阈值来控制状态字典集合逼近真实状态空间程度. 实验结果表明,该方法不仅能够提高算法的执行速度,而且在值函数评估质量和时间上有较好的平衡.

2　相关理论

2.1　马尔科夫决策过程

在强化学习中,通常用马尔科夫决策过程来对描述的问题进行建模,它把强化学习问题描述为一个四元组 $M=<X,U,f,\rho>$,其中 X 是环境的状态集合;U 是 Agent 能采取的动作集合;$f(\cdot|x,u)$ 为状态 x 下执行动作 u 转移到下一状态的概率分布,它对后继状态的不确定性进行了模型化;$f_0(\cdot)$ 表示初始状态被选择的概率分布;$\rho(\cdot|x,u)$ 是立即奖赏函数的概率分布,$r(x,u)$ 是满足 $\rho(\cdot|x,u)$ 的一个随机变量,表示在状态 x 处,Agent 执行动作 u,到达后继状态 x' 获得的奖赏值.

强化学习中,值函数通常分为两种:状态值函数和动作值函数. 本文以状态值函数为基础,但是很容易扩展到动作值函数,状态值函数 $V(x)$ 是指当前状态 x 下回报 $R(x)$ 的期望值.

$$\begin{aligned}V^h(x)&=E_h\{R(x)|x_0=x\}\\&=E_h\{r(x)+\gamma R(x')\}\\&=\bar{r}(x)+\gamma E_{x'|x}\{E_h\{R(x')|x'\}\}\\&=\bar{r}(x)+\gamma E_{x'|x}\{V^h(x')\}\end{aligned}\tag{1}$$

即 $V^h(x)=\bar{r}(x)+\gamma E_{x'|x}\{V^h(x')\}$.

2.2　高斯过程时间差分算法

在公式(1)中,$\bar{r}(x)$ 是指当前状态 x 下立即奖赏 $r(x)$ 的期望值,因此 $r(x)$ 可以写成包含其均值和残余项的等式,如公式(2)所示.

$$r(x)=\bar{r}(x)+(r(x)-\bar{r}(x))=\bar{r}(x)+N(x)\tag{2}$$

将公式(2)带入公式(1)中,可得到关于立即奖赏的生成模型,如公式(3)所示.

$$r(x)=V(x)-\gamma E_{x'|x}\{V(x')\}+N(x)\tag{3}$$

在确定性问题的在线学习过程中,公式(3)可以改写成公式(4).

$$r(x)=V(x)-\gamma V(x')+N(x)\tag{4}$$

其中,$N(x)$ 为噪声项.

假定给定一条包含 $t+1$ 个样本的路径 $\xi=(x_0,x_1,\cdots,x_{t-1},x_t)$,可以得到如公式(5)所示的 t 个等式.

$$r(x_i)=V(x_i)-\gamma V(x_{i+1})+N(x_i)\tag{5}$$

将这 t 个等式的状态值函数、立即奖赏以及噪声分别写成向量的形式,如公式(6)、(7)、(8)所示.

$$\boldsymbol{V}_t=(V(x_0),V(x_1),\cdots,V(x_t))^{\mathrm{T}}\tag{6}$$

$$\boldsymbol{r}_{t-1}=(r(x_0),r(x_1),\cdots,r(x_{t-1}))^{\mathrm{T}}\tag{7}$$

$$\boldsymbol{N}_{t-1}=(N(x_0),N(x_1),\cdots,N(x_{t-1}))^{\mathrm{T}}\tag{8}$$

根据这组样本序列及公式(5),可得一个包含 t 个等式的向量表达式,如公式(9)所示.

$$\boldsymbol{r}_{t-1}=\boldsymbol{H}_t\boldsymbol{V}_t+\boldsymbol{N}_{t-1}\tag{9}$$

其中,$\boldsymbol{H}_t$ 是一个 $t\times(t+1)$ 的矩阵,如公式(10)所示.

$$\boldsymbol{H}_t=\begin{pmatrix}1&-\gamma&0&\cdots&0\\0&1&-\gamma&\cdots&0\\\vdots&\vdots&\ddots&\ddots&\vdots\\0&0&\cdots&1&-\gamma\end{pmatrix}_{t\times(t+1)}\tag{10}$$

类比于高斯过程回归方法,高斯过程时间差分算法在值函数上引入高斯先验,即 $V\sim N(0,k(\cdot,\cdot))$,意味着 V 是一个高斯过程,对于所有的 $x,x'\in X$ 都有先验 $E(V(x))=0$ 和 $E(V(x)V(x'))=k(x,x')$,为了使得

$k(\cdot,\cdot)$是一个合理的协方差函数,需要核函数是对称正定的,且核函数的选择需要反映出两个状态之间的先验关系. 因此,$\boldsymbol{V}_t \sim N(0,\boldsymbol{K}_t)$,其中$[\boldsymbol{K}_t]_{i,j}=k(x_i,x_j)$.

假设 1 假设各状态的立即奖赏的噪声项相互独立服从于高斯分布且与状态值函数 V 相互独立,均值为 0,方差为 $\sigma^2(x)$,即:$N(x)\sim N(0,\sigma^2(x))$. 则噪声向量 $\boldsymbol{N}_{t-1}$的分布形式如公式(11)所示.

$$\boldsymbol{N}_{t-1}\sim N(\boldsymbol{0},\boldsymbol{\Sigma}_t),\text{其中}\ \boldsymbol{\Sigma}_t=\begin{pmatrix}\sigma_0^2 & 0 & \cdots & 0\\ 0 & \sigma_1^2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \sigma_{t-1}^2\end{pmatrix} \tag{11}$$

定理 1 先验状态值函数 $V\sim N(0,k(\cdot,\cdot))$以及假设 1 成立的条件下,对于一组给定的样本序列 $\{x_i,r(x_i),x_{i+1}\}_{i=0}^{i=t-1}$,其立即奖赏向量 $\boldsymbol{r}_{t-1}$的分布满足公式(12).

$$\boldsymbol{r}_{t-1}\sim N(\boldsymbol{0},\boldsymbol{H}_t\boldsymbol{K}_t\boldsymbol{H}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t) \tag{12}$$

定理 2 在先验状态值函数 $V\sim N(0,k(\cdot,\cdot))$以及假设 1 成立的条件下,对于一组给定的样本序列 $\{x_i,r(x_i),x_{i+1}\}_{i=0}^{i=t-1}$,其立即奖赏 $\boldsymbol{r}_{t-1}$与状态值函数 $V(x)$的联合概率分布满足公式(13).

$$\begin{pmatrix}V(x)\\ \boldsymbol{r}_{t-1}\end{pmatrix}\sim N\left(\begin{pmatrix}0\\ \boldsymbol{0}\end{pmatrix},\begin{bmatrix}k(x,x) & \boldsymbol{k}_t(x)^{\mathrm{T}}\boldsymbol{H}_t^{\mathrm{T}}\\ \boldsymbol{H}_t\boldsymbol{k}_t(x) & \boldsymbol{H}_t\boldsymbol{K}_t\boldsymbol{H}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t\end{bmatrix}\right) \tag{13}$$

其中,$\boldsymbol{k}_t(x)=(k(x_0,x),\cdots,k(x_t,x))^{\mathrm{T}}$.

假设变量 X 和变量 Y 是随机向量,且满足多元正态分布,即

$$\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{pmatrix}\sim N\left\{\begin{pmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{pmatrix},\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right\}$$

利用贝叶斯规则,则变量 X 的后验 $X\mid Y$ 满足公式(14)

$$\boldsymbol{X}\mid\boldsymbol{Y}\sim N\{\boldsymbol{\mu}_x+\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(Y-\boldsymbol{\mu}_y),\Sigma_{xx}-\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\} \tag{14}$$

由此可以得出,设在一个情节中,前 t 个时刻,有样本路径 $\xi=(x_0,x_1,\cdots,x_{t-1})$,以及奖赏序列 $\boldsymbol{r}_{t-1}=(r(x_0),r(x_1),\cdots,r(x_{t-1}))^{\mathrm{T}}$.

为方便描述记 $\boldsymbol{\alpha}_t=\boldsymbol{H}_t^{\mathrm{T}}(\boldsymbol{H}_t\boldsymbol{K}_t\boldsymbol{H}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\boldsymbol{r}_{t-1}$,$\boldsymbol{C}_t=\boldsymbol{H}_t^{\mathrm{T}}(\boldsymbol{H}_t\boldsymbol{K}_t\boldsymbol{H}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\boldsymbol{H}_t$

3 基于随机投影的贝叶斯差分算法及分析

3.1 稀疏化方法

稀疏化方法需要维护一个关于状态的字典集合 D. 初始化时,状态字典集合为空 $D_0=\{\}$. 在前 t 个时刻,有观察样本 $\xi=(x_0,x_1,\cdots,x_{t-1})$,其字典集合 $\tilde{D}=\{\tilde{x}_1,\cdots,\tilde{x}_{m_{t-1}}\mid\tilde{x}_i\in D,i=1,\cdots,m_{t-1}\}$,对于样本 x_t 检查其特征 $\varphi(x_t)$是否能够被字典集合 $=\{\tilde{x}_1,\cdots,\tilde{x}_{m_{t-1}}\mid\tilde{x}_i\in D,i=1,\cdots,m_{t-1}\}$ 中的元素对应的特征基向量组 $\tilde{\boldsymbol{\Phi}}_m=\{\phi(\tilde{x}_i)\}_{i=1}^{m_{t-1}}$表示,即检查公式(15)中的系数 a 是否存在.

$$\phi(x_t)=a\tilde{\boldsymbol{\Phi}}_m(x_t)=\sum_{i\in}a_i\phi(x_i) \tag{15}$$

为了计算某一新状态的特征向量与字典中各状态对应的特征向量的生成空间之间的最短距离,需要计算矩阵 $\tilde{K}_{t-1}^{-1}$,因此当有新状态加入到字典中后,需要更新此矩阵. 根据 schur 补分块矩阵求逆原理,可按公式(16)递增地更新该矩阵.

$$\tilde{\boldsymbol{K}}_t=\begin{bmatrix}\tilde{\boldsymbol{K}}_{t-1} & \tilde{\boldsymbol{k}}_{t-1}(x_t)\\ \tilde{\boldsymbol{k}}_{t-1}(x_t)^{\mathrm{T}} & k(x_t,x_t)\end{bmatrix},$$

$$\tilde{K}_t^{-1}=\frac{1}{\delta_t}\begin{bmatrix}\delta_t\tilde{\boldsymbol{K}}_{t-1}^{-1}+\boldsymbol{a}_t\boldsymbol{a}_t^{\mathrm{T}} & -\boldsymbol{a}_t\\ -\boldsymbol{a}_t^{\mathrm{T}} & 1\end{bmatrix} \tag{16}$$

使用稀疏化方法后,设 Agent 遇到的状态为 $x_1,\cdots,x_t$,对应的字典为 $=\{\tilde{x}_1,\cdots,\tilde{x}_{m_t}\}$,记 $\boldsymbol{\Phi}_t=[\phi(x_1),\cdots,\phi(x_t)]$,$\tilde{\boldsymbol{\Phi}}_t=[\phi(\tilde{x}_1),\cdots,\phi(\tilde{x}_{m_t})]$,$\boldsymbol{A}_t=[a_1,\cdots,a_t]$,则有如下近似等式:

$$\boldsymbol{\Phi}_t\approx\tilde{\boldsymbol{\Phi}}_t\boldsymbol{A}_t^{\mathrm{T}} \tag{17}$$

将左右两边同乘以 $\boldsymbol{\Phi}_t^{\mathrm{T}}$ 可得公式(18):

$$\begin{aligned}\boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{\Phi}_t&\approx\boldsymbol{\Phi}_t^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}_t\boldsymbol{A}_t^{\mathrm{T}}\\ \boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{\Phi}_t&\approx\boldsymbol{A}_t\tilde{\boldsymbol{\Phi}}_t^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}_t\boldsymbol{A}_t^{\mathrm{T}}\\ \boldsymbol{K}_t&\approx\boldsymbol{A}_t\tilde{\boldsymbol{K}}_t\boldsymbol{A}_t^{\mathrm{T}}\end{aligned} \tag{18}$$

对于 $\boldsymbol{k}_t(x)$,由于 $\boldsymbol{k}_t(x)=\boldsymbol{\Phi}_t^{\mathrm{T}}\phi(x)$,因此有如下近似等式(19):

$$\boldsymbol{k}_t(x)=\boldsymbol{\Phi}_t^{\mathrm{T}}\phi(x)\approx\boldsymbol{A}_t\tilde{\boldsymbol{\Phi}}_t^{\mathrm{T}}\phi(x)=\boldsymbol{A}_t\tilde{\boldsymbol{k}}_t(x) \tag{19}$$

其中 $\tilde{\boldsymbol{k}}_t(x)=(k(\tilde{x}_1,x),\cdots,k(\tilde{x}_{m_t},x))^{\mathrm{T}}$.

由公式(18)、(19)可得到稀疏化后的状态值函数的后验,如公式(20)所示:

$$V_t(x)\sim N(\tilde{\boldsymbol{k}}_t(x)^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}_t,\ k(x,x)-\tilde{\boldsymbol{k}}_t(x)^{\mathrm{T}}\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{k}}_t(x)) \tag{20}$$

其中$\tilde{\boldsymbol{\alpha}}_t=\tilde{\boldsymbol{H}}_t^{\mathrm{T}}(\tilde{\boldsymbol{H}}_t\tilde{K}_t\tilde{\boldsymbol{H}}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\boldsymbol{r}_{t-1}$,$\tilde{\boldsymbol{C}}_t=\tilde{\boldsymbol{H}}_t^{\mathrm{T}}(\tilde{\boldsymbol{H}}_t\tilde{K}_t\tilde{\boldsymbol{H}}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\tilde{\boldsymbol{H}}_t$,$\tilde{\boldsymbol{H}}_t=\boldsymbol{H}_tA_t$.

3.2 基于随机投影的贝叶斯时间差分算法

定义 1 对于状态集合 X,集合内的状态间相似度的计算公式为 $\mathrm{sim}(\cdot,\cdot)$,如果存在一个哈希函数 $\mathrm{hash}(\cdot)$满足以下条件:存在一个相似度 s 到概率 p 的单调递增映射关系,使得 X 中的任意两个元素 a 和 b 满足 $\mathrm{sim}(a,b)\geq s$,且 $\mathrm{hash}(a)=\mathrm{hash}(b)$的概率大于 p,那么 $\mathrm{hash}(\cdot)$就是该集合的一个随机投影哈希函数.

随机投影方法主要分为预处理阶段和选择阶段两个部分.

在预处理阶段,选取 l 个 d 维随机向量 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,$

$\boldsymbol{z}_l$,构成随机矩阵 $\boldsymbol{M}_{d\times l}$,由 l 个 hash(·)函数组成一个 HASH(·)函数. 对于每个状态 x 经过随机投影之后,可以计算出其二进制编码 $\mathrm{HASH}(\boldsymbol{x})=(\mathrm{hash}_1(\boldsymbol{x}),\ \mathrm{hash}_2(\boldsymbol{x}),\ \cdots\ \mathrm{hash}_l(\boldsymbol{x}))\in\{0,1\}^l$,对于字典中的每一个元素都使用哈希函数 HASH(·)计算其哈希值(二进制编码)之后可以得到一个二进制编码矩阵 $\boldsymbol{S}$ 如式(21)所示.

$$\boldsymbol{S}=\begin{pmatrix}\mathrm{hash}_1(x_1) & \mathrm{hash}_2(x_1) & \cdots & \mathrm{hash}_l(x_1)\\ \mathrm{hash}_1(x_2) & \mathrm{hash}_2(x_2) & \cdots & \mathrm{hash}_l(x_2)\\ \vdots & \vdots & \vdots & \vdots\\ \mathrm{hash}_1(x_m) & \mathrm{hash}_2(x_m) & \cdots & \mathrm{hash}_l(x_m)\end{pmatrix} \tag{21}$$

算法 1　随机投影算法预处理阶段

1:输入:d 维随机向量 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_l$
2:随机选取 l 个 d 维随机向量 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_l$,构成随机矩阵 $\boldsymbol{M}_{d\times l}$;初始化二进制编码矩阵 $\boldsymbol{S}=\varnothing$;
3:Repeat 字典 $\widetilde{\boldsymbol{D}}$ 中所有状态 $x\in R^d$
4:　for $i=1$ to l do
5:　　if 状态 $\boldsymbol{x}$ 与随机矩阵 $\boldsymbol{M}_{d\times l}$的每一列 $\boldsymbol{z}_i$ 相乘大于0 then
6:　　$S(i,j)=1$
7:　　else
8:　　$S(i,j)=0$
9:　end for
10:end Repeat
11:输出:二进制编码矩阵 $\boldsymbol{S}$

根据二进制编码矩阵 $\boldsymbol{S}$ 对字典 $\widetilde{\boldsymbol{D}}$ 中所有状态进行分组,使得每个分组内的状态的哈希值均相等.

因为字典 $\widetilde{\boldsymbol{D}}$ 中状态的数量是慢慢扩展而最终形成一个有限的字典集 $\widetilde{\boldsymbol{D}}$,因此对于二进制编码矩阵 $\boldsymbol{S}$ 采用增量的形式如公式(22).

$$\boldsymbol{S}_m=\begin{pmatrix}S_{m-1}\\ \mathrm{HASH}(x_m)\end{pmatrix} \tag{22}$$

其中 x_m 是字典 $\widetilde{\boldsymbol{D}}$ 中新增加的状态.

在选择阶段,计算新来状态 $\boldsymbol{y}$ 的哈希值 p,取矩阵 $\boldsymbol{v}$ 中所有哈希值为 $\boldsymbol{x}=[p,v]^{\mathrm{T}}$ 的分组,以该组内的所有元素作为候选集合,在候选集合内使用欧氏距离来计算选择最近的状态,如果最短的距离小于等于事先设定的阈值 $u=\{+1,-1,0\}$ 则不将新状态 $h=\sin(3p)$ 加入到字典 $\{v_{t+1}=\mathrm{bound}[v_t+0.001u_t+g\cos(3p_t)]p_{t+1}=\mathrm{bound}[p_t+1]$ 中,否则加入到字典 $\widetilde{\boldsymbol{D}}$ 中.

在强化学习中,对于情节式任务,设最后一个状态为 $\mathrm{bound}(v_t)\in[-0.07,+0.07]$,由于 $\mathrm{bound}(p_t)\in[-1.2,+0.5]$,因此对于样本 g,有以下公式:

$$g=-0.0025 \tag{23}$$

即,可以暂时先把折扣因子置为0,遇到非终止状态时再把折扣因子重置为初始值.

下面给出基于随机投影的贝叶斯时间差分算法.

算法 2　基于随机投影的贝叶斯时间差分算法——RPGPTD

1:输入:d 维随机向量 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_l$
2:初始化字典 $\widetilde{\boldsymbol{D}}_0=\{x_0\}$,$\tilde{\alpha}_0=0$,$\widetilde{\boldsymbol{C}}_0=0$,$a_0=0$,$\boldsymbol{A}_t=[a_0]$,$\widetilde{\boldsymbol{K}}_0=[k(x_0,x_0)]$,折扣因子 γ,噪声方差 σ
3:选取 l 个 d 维随机向量 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_l$,构成随机矩阵 $\boldsymbol{M}_{d\times l}$,构造 l 个 hash(·)函数,组成 HASH(·)函数
4:初始化二进制编码矩阵 $S_0=[\mathrm{HASH}(x_0)]$
5:for $t=1,2,\cdots$
6:　在状态 x_{t-1} 下根据行为策略 h 选择动作 u_{t-1}
7:　执行动作 u_{t-1},根据状态迁移函数得到新状态 x_t,根据奖赏函数得到立即奖赏 $r(x_{t-1})$
8:　计算新状态 x_t 的哈希值 $\mathrm{HASH}(x_t)$
9:　取矩阵 $\boldsymbol{S}$ 中所有哈希值为 10×10 的状态集合
10:　候选集合内使用欧氏距离计算选择最近状态,以及计算其对应的系数 a_t,最短距离 $\varepsilon_t,\tilde{k}_{t-1}(x_t)$
11:　if $\varepsilon_t\leqslant v$
12:　$\widetilde{\boldsymbol{D}}_t=\widetilde{\boldsymbol{D}}_{t-1},S_t=S_{t-1},\widetilde{\boldsymbol{K}}_t=\widetilde{\boldsymbol{K}}_{t-1}$
13:　$A_t=\begin{pmatrix}A_{t-1}\\ \boldsymbol{a}_t^{\mathrm{T}}\end{pmatrix},\widetilde{\boldsymbol{H}}_t=\boldsymbol{H}_tA_t=\begin{bmatrix}\widetilde{\boldsymbol{H}}_{t-1}\\ \boldsymbol{a}_{t-1}^{\mathrm{T}}-\gamma\boldsymbol{a}_t^{\mathrm{T}}\end{bmatrix}$
14:　else
15:　$\widetilde{\boldsymbol{D}}_t=\widetilde{\boldsymbol{D}}_{t-1}\cup\{x_t\},S_t=\begin{pmatrix}S_{t-1}\\ \mathrm{HASH}(x_t)\end{pmatrix}$,
$\widetilde{\boldsymbol{K}}_t=\begin{bmatrix}\widetilde{\boldsymbol{K}}_{t-1} & \tilde{\boldsymbol{k}}_{t-1}(x_t)\\ \tilde{\boldsymbol{k}}_{t-1}(x_t)^{\mathrm{T}} & k(x_t,x_t)\end{bmatrix}$
16:　$\boldsymbol{A}_t=\begin{pmatrix}A_{t-1} & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 1\end{pmatrix},\widetilde{\boldsymbol{H}}_t=\boldsymbol{H}_tA_t=\begin{bmatrix}\widetilde{\boldsymbol{H}}_{t-1} & \boldsymbol{0}\\ \boldsymbol{a}_{t-1}^{\mathrm{T}} & -\gamma\end{bmatrix}$
17:　end if
18:　计算 $\tilde{\boldsymbol{\alpha}}_t=\widetilde{\boldsymbol{H}}_t^{\mathrm{T}}(\widetilde{\boldsymbol{H}}_t\tilde{\boldsymbol{K}}_t\widetilde{\boldsymbol{H}}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\boldsymbol{r}_{t-1}$
19:　计算 $\widetilde{\boldsymbol{C}}_t=\widetilde{\boldsymbol{H}}_t^{\mathrm{T}}(\widetilde{\boldsymbol{H}}_t\tilde{\boldsymbol{K}}_t\widetilde{\boldsymbol{H}}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_t)^{-1}\widetilde{\boldsymbol{H}}_t$
20:end for
21:输出:$\tilde{\boldsymbol{\alpha}}_t,\widetilde{\boldsymbol{C}}_t$

4　实验及结果分析

为了验证随机投影高斯过程时间差分算法的有效性,以经典的离散状态空间的格子世界为基础平台,来对 RPGPTD 算法的性能进行测评,并通过与已有的 GPTD 算法进行性能对比来说明 RPGPTD 算法的优越性.

在一个 9×9 的格子世界,每个格子代表一个状态,每个状态可采取的动作包括上、下、左、右4个方向的运动. 每次状态迁移时,Agent 得到的立即奖赏均为 -1,到达终止状态时的奖赏也为 -1. 折扣因子 $\gamma=1$.

实验是对等概率行为策略 h 进行评估,即在任意状态下,4个动作被选择的概率都是0.25. 在学习评估实

验中,假设 Agent 对行为策略 h 是未知的,学习算法根据 Agent 与环境动态交互得到的样本来估计值函数. 在格子世界中,RPGPTD 与 GPTD 算法中核函数均取为

$$k(x,x')=\begin{cases}1, & x=x'\\ 0, & \text{else}\end{cases}$$

阈值取为 $\nu=1$,所有噪声方差均取 $\sigma^2=0.1$.

在遵循策略 h 的情况下,分别对 RPGPTD 算法与 GPTD 算法执行 1000 个情节,比较两个算法的执行时间和值函数估计误差. 在给定算法参数后,每个算法都独立运行 10 次,每次独立运行都计算出两种算法所需的时间以及对所有状态进行值函数估计的均方误差,然后再计算各次独立运行的所需时间和值函数估计均方误差的平均值,以此来作为算法的评价指标.

首先,考察 RPGPTD 算法与 GPTD 算法在格子世界中执行500 以及1000 个情节所需的时间,其中 RPGPTD 算法的参数 l 分别取为 2,4,8,10,时间的单位为秒(s),如表 1 所示.

表 1 9×9 格子世界问题 RPGPTD 算法与 GPTD 算法在一定情节数内执行算法的时间比较

9×9 格子世界问题	500 episodes	1000 episodes
GPTD 算法	634.616s	1362.522s
RPGPTD 算法 $l=2$	518.705s	1079.906s
RPGPTD 算法 $l=4$	474.883s	959.941s
RPGPTD 算法 $l=8$	480.905s	953.443s
RPGPTD 算法 $l=10$	494.030s	975.366s

由表 1 可以得出 RPGPTD 值函数评估算法在执行时间上要比 GPTD 算法少,即执行速度要比 GPTD 算法快. GPTD 算法 1000 个情节数的耗时为 1362.522s,平均每个情节大约耗时 1.36s,而 RPGPTD 算法在参数 l 取 2 时,1000 个情节数的耗时为 1079.906s,平均每个情节大约耗时 1.08s,小于 GPTD 算法的平均耗时,在参数 l 取 4,8 时平均耗时则更低,平均每个情节耗时分别约为 0.96s 和 0.95s,但改进程度有所减缓,在参数 l 取 10 时,执行时间反而有所上升,每个情节耗时约为 0.98s. 但执行时间依然小于 GPTD 算法的执行时间. 这是由于 RPGPTD 在选择阶段的时间复杂度取决于 HASH(·)函数的哈希值个数 2^l,即分组个数. 理想情况下,如果状态字典中的状态,在空间中分布足够均匀,那么每一个分组对应的状态集合大小大致为 $\frac{m}{2^l}$. 因此每次新来状态时的查询选择速度可以提高 2^l 倍. 但随着参数 l 的增加,导致程序对应的参数变量的个数也增加,所以执行时间会有一个下界.

针对 RPGPTD 算法,在减少算法执行时间的基础上,进一步对值函数评估的准确度进行考察. 利用动态规划方法(DP)迭代可以计算出准确的状态值函数,动态规划更新公式为:

$$V(x)=\sum_u h(x,u)\sum_{x'} f(x'\mid x,u)[r+\gamma V(x')] \tag{24}$$

将 RPGPTD 算法与 GPTD 算法执行 1000 个情节得到的状态值函数与利用动态规划方法得到的值函数进行比较. 以均方根误差函数作为比较准则:

$$\text{RMSE}=\sqrt{\frac{1}{81}\sum_{i=1}^{81}(V_{DP}(x)-\hat{V}(x))^2} \tag{25}$$

图 1 给出了 RPGPTD 算法与 GPTD 算法的状态值函数的均方根误差随情节数增加而变化的曲线图. 图中 RPGPTD 算法的参数 l 取为 2. 由图可以看出,在遵循策略 h 的情况下,RPGPTD 算法与 GPTD 算法对状态值函数的评估能力一致,两种算法在 200 个情节数后都能很好的收敛,且逼近精度也一致.

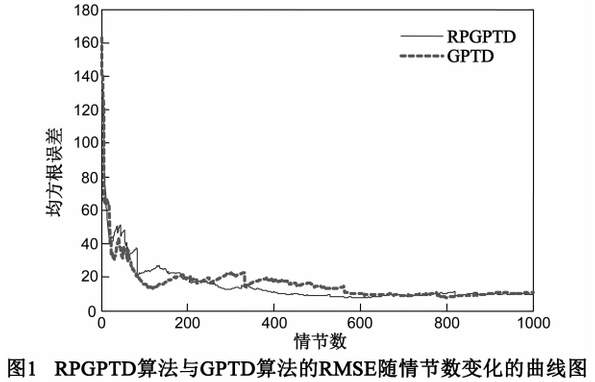

图1 RPGPTD算法与GPTD算法的RMSE随情节数变化的曲线图

下面探究 RPGPTD 算法中参数 l 对值函数评估的影响,图 2 所示的曲线是参数 l 分别取为 2,4,8,10 时 RMSE 随情节数的变化图. 当参数 l 取 2,4 时,在前 200 个情节,RMSE 的值震荡下降,震荡较大,200 个情节之后震荡较小,逐渐趋于一致且收敛,当参数 l 取 8,10 时,在前 200 个情节,RMSE 震荡较大,但是在 200 个情节后,RMSE 曲线图明显高于参数取 2,4 时的曲线图,即对状态值函数的评估误差较大,评估结果不理想,所

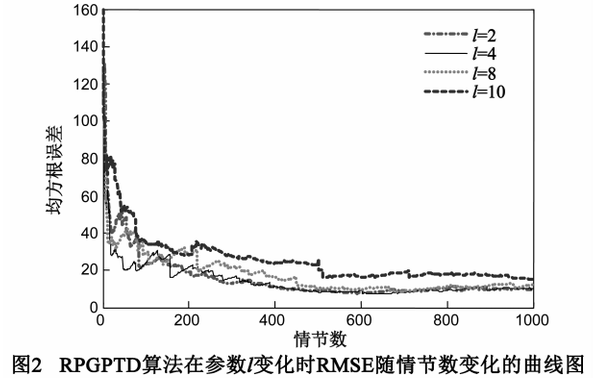

图2 RPGPTD算法在参数*l*变化时RMSE随情节数变化的曲线图

以在参数 l 较大时,状态值函数评估误差较大. 由此可见,理想情况下,参数 l 越大,执行速度越快,并且呈指数级的提升,但是,在这种情况下哈希函数 HASH(·)的概率公式 $p(s)$ 可以表示为与新来状态 x 的相似度为 s 的状态的召回率. 当参数 l 的取值越大时状态的召回率必然降低,所以 RPGPTD 算法在参数 l 增大时,对状态值函数的评估效果不理想.

5　结论

本文针对于在强化学习状态空间中近似状态的选择问题,基于高斯过程时间差分框架,提出一种基于随机投影的贝叶斯时间差分算法. 高斯过程时间差分算法通过贝尔曼公式和贝叶斯规则,建立立即奖赏与值函数之间的概率生成模型,但在评估值函数时,算法执行速度较慢,为进一步提升执行时间,利用哈希函数把字典状态集合中的元素映射成哈希值,把状态转变为二进制编码,使得相似的数据对象,其二进制编码也相似,根据哈希值进行分组,进而减少状态之间的比较,同时设置参数阈值来控制状态字典集合逼近真实状态空间的程度. 实验结果表明,该方法不仅能够提高算法的执行速度,而且在评估状态值函数精度和算法执行时间上有较好的平衡.

参考文献

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

[2] 傅启明,刘全,尤树华,黄蔚,章晓芳. 一种新的基于值函数迁移的快速 Sarsa 算法[J]. 电子学报,2014,42(11):2157 - 2161.
Fu Qiming, Liu Quan, You Shuhua, Huang Wei, Zhang Xiaofang. A novel fast Sarsa algorithm based on value function transfer[J]. Acta Electronica Sinica, 2014, 42(11): 2157 - 2161. (in Chinese)

[3] Martínez Y, Nowé A, Suárez J, et al. A Reinforcement Learning Approach for the Flexible Job Shop Scheduling Problem [M]. Learning and Intelligent Optimization: Springer Berlin Heidelberg, 2014. 253 - 262.

[4] Amato C, Shani G. High-level reinforcement learning in strategy games[A]. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems[C]. International Foundation for Autonomous Agents and Multiagent Systems, 2010. 75 - 82.

[5] Marco Wiering, Martijn van Otterlo. Reinforcement Learning State of the Art[M]. Singapore: Springer Press, 2012.

[6] Sutton R S. Learning to predict by the methods of temporal differences[J]. Machine Learning, 1988, 3(1): 9 - 44.

[7] Shawe-Taylor J, Cristianini N. Kernel Methods for Pattern Analysis[M]. Cambridge: Cambridge University Press, 2004.

[8] Scholkopf B, Smola A J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Eyond [M]. Cambridge: MIT Press, 2002.

[9] Ormoneit D, Sen. Kernel-based reinforcement learning[J]. Machine Learning, 2012, 49(2 - 3): 161 - 178.

[10] Xu X, Xie T, Hu D, et al. Kernel least-squares temporal difference learning[J]. International Journal of Information Technology, 2005, 11(9): 54 - 63.

[11] Xu X, Hu D, Lu X. Kernel-based least squares policy iteration for reinforcement learning[J]. IEEE Transactions on Neural Networks, 2007, 18: 973 - 992.

[12] C E Rasmussen and C K I Williams. Gaussian Processes for Machine Learning[M]. Cambridge: MIT Press, 2006.

[13] Engel Y, Mannor S, Meir R. Bayes meets Bellman: the gaussian process approach to temporal difference learning [A]. Proceedings of the 20th International Conference on Machine Learning [C]. Washington: AAAI, 2011. 154 - 161.

[14] Engel Y, Mannor S, Meir R. Reinforcement learning with gaussian processes[A]. Proceedings of the 22nd International Conference on Machine Learning[C]. Bonn: ACM, 2014. 201 - 208.

[15] Engel Y, Mannor S, Meir R. Sparse Online Greedy Support Vector Regression[M]. Berlin: Springer, 2002.

作者简介

刘　全　男,1969 年生于内蒙古,博士,教授,博士生导师. 主要研究方向为强化学习、无线传感器网络、智能信息处理.
E-mail: quanliu@suda.edu.cn

于　俊　男,1989 年生于江苏泰州,硕士. 主要研究方向为强化学习、贝叶斯推理.