强化学习研究进展*

1 引言

智能系统的一个主要特征是能够适应未知环境，其中学习能力是智能系统的关键技术之一。在机器学习范畴，根据反馈的不同，学习技术可以分为监督学习（Supervised learning）、非监督学习（Unsupervised learning）和强化学习（Reinforcement learning）三大类。其中强化学习是一种以环境反馈作为输入的、特殊的、适应环境的机器学习方法。所谓强化学习是指从环境状态到行为映射的学习，以使系统行为从环境中获得的累积奖赏值最大。该方法不同与监督学习技术那样通过正例、反例来告知采取何种行为，而是通过试错（trial-and-error）的方法来发现最优行为策略[KLM96][SB98]。

强化学习通常包括两个方面的含义：一方面是将强化学习作为一类问题；另一方面是指解决这类问题的一种技术。如果将强化学习作为一类问题，目前的学习技术大致可分成两类：其一是搜索智能系统的行为空间，以发现系统最优的行为。典型的技术如遗传算法等搜索技术；另一类是采用统计技术和动态规划方法来估计在某一环境状态下的行为的效用函数值，从而通过行为效用函数来确定最优行为。我们特指这种学习技术为强化学习技术。不作特殊说明，在本章中强化学习被理解为是一种学习技术。

强化学习技术是从控制理论、统计学、心理学等相关学科发展而来，最早可以追溯到巴普洛夫的条件反射实验。但直到上世纪八十年代末、九十年代初强化学习技术才在人工智能、机器学习和自动控制等领域中得到广泛研究和应用，并被认为是设计智能系统的核心技术之一。特别是随着强化学习的数学基础研究取得突破性进展后，对强化学习的研究和应用日益开展起来，成为目前机器学习领域的研究热点之一。

本章综述了强化学习技术这一领域的研究情况，特别是从第 3 节至第 6 节讨论了当前强化学习研究中的热点问题。第 2 节简要介绍典型强化学习算法及其数学基础，第 3 节介绍部分感知环境下的强化学习算法，第 4 节介绍强化学习中连续状态的函数估计，第 5 节介绍分层强化学习技术，第 6 节介绍多 agent 强化学习研究，最后在第 7 节进行了总结和展望。

2 强化学习基础

一个智能系统面临的环境往往是动态、复杂的开放环境。因此首先需要设计者对环境加以细分。通常情况，我们可以从以下五个角度对环境（或问题）进行分析[Ger99]。

表 1　环境的描述

角度 1	离散状态 vs 连续状态
角度 2	状态完全可感知 vs 状态部分可感知

* 本文得到国家自然科学基金(60475026)和江苏省创新人才项目(BK2003409)资助

角度 3	插曲式 vs 非插曲式
角度 4	确定性 vs 不确定性
角度 5	静态 vs 动态

表 1 中，所谓插曲式（episodic）是指智能系统在每个场景中学习的知识对下一个场景中的学习是有用的。如一个棋类程序对同一个对手时，在每一棋局中学习的策略对下一棋局都是有帮助的。相反非插曲式（non-episodic）环境是指智能系统在不同场景中学习的知识是无关的。角度 4 是指智能系统所处的环境中，如果状态的迁移是确定的，则可以唯一确定下一状态。否则在不确定性环境中，下一状态是依赖于某种概率分布。进一步，如果状态迁移的概率模型是稳定、不变的，则称之为静态环境；否则为动态环境。显然，最复杂的一类环境（或问题）是连续状态、部分可感知、非插曲式、不确定的动态环境。

在强化学习技术中首先对随机的、离散状态、离散时间这一类问题进行数学建模。在实际应用中，最常采用的是马尔可夫模型。表 2 中给出最常用的几种马氏模型。

表 2 常用的几种马氏模型

马氏模型		是否智能系统行为控制环境状态转移？	
		否	是
是否环境为部分可感知？	否	马尔可夫链	马氏决策过程
	是	隐马尔可夫模型	部分感知马氏决策过程

基于表 2 中的马氏决策过程，强化学习可以简化为图 1 的结构。图 1 中，强化学习系统接受环境状态的输入 s，根据内部的推理机制，系统输出相应的行为动作 a。环境在系统动作作用 a 下，变迁到新的状态 s'。系统接受环境新状态的输入，同时得到环境对于系统的瞬时奖惩反馈 r。对于强化学习系统来讲，其目标是学习一个行为策略 $\pi: S\to A$，使系统选择的动作能够获得环境奖赏的累计值最大。换言之，系统要最大化（1）式，其中 γ 为折扣因子。在学习过程中，强化学习技术的基本原理是：如果系统某个动作导致环境正的奖赏，那么系统以后产生这个动作的趋势便会加强。反之系统产生这个动作的趋势便减弱。这和生理学中的条件反射原理是接近的。

$$\sum_{i=0}^{\infty}\gamma^{i}r_{t+i} \qquad 0<\gamma\le 1 \tag{1}$$

如果假定环境是马尔可夫型的，则顺序型强化学习问题可以通过马氏决策过程（Markov Decision Process，MDP）建模。下面首先给出马氏决策过程的形式化定义[KLM96]。

马氏决策过程 由四元组<S，A，R，P>定义。包含一个环境状态集 S，系统行为集合 A，奖赏函数 $R: S\times A\to\mathcal{R}$ 和状态转移函数 $P: S\times A\to PD(S)$。记 $R(s, a, s')$ 为系统在状态 s 采用 a 动作使环境状态转移到 s'获得的瞬时奖赏值；记 $P(s, a, s')$ 为系统在状态 s 采用 a 动作使环境状态转移到 s'的概率。[1]

马氏决策过程的本质是：当前状态向下一状态转移的概率和奖赏值只取决于当前状态和选择的动作，而与历史状态和历史动作无关。因此在已知状态转移概率函数 P 和奖赏函数 R 的环境模型知识下，可以采用动态规划技术求解最优策略。而强化学习着重研究在 P 函数和 R

[1] 在下文中也分别标记为 $R_{ss'}^{a}$ 和 $P_{ss'}^{a}$。

函数未知的情况下，系统如何学习最优行为策略。

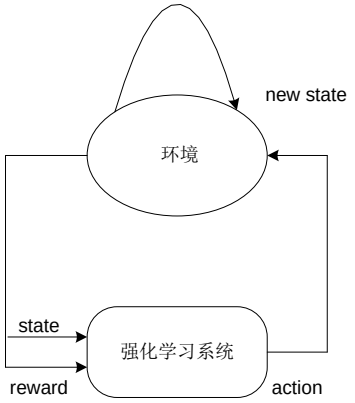


图 1 强化学习结构

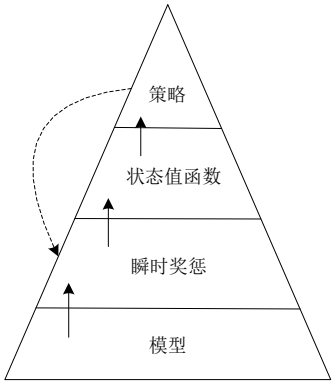


图 2 强化学习四要素

为解决这个问题，图 2 中给出强化学习四个关键要素之间的关系，四要素关系自底向上呈金字塔结构。系统所面临的环境由环境模型定义，但由于模型中 P 函数和 R 函数未知，系统只能够依赖于每次试错所获得的瞬时奖赏来选择策略。但由于在选择行为策略过程中，要考虑到环境模型的不确定性和目标的长远性，因此在策略和瞬时奖赏之间构造值函数（即状态的效用函数），用于策略的选择。

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = r_{t+1} + \gamma R_{t+1} \tag{2}$$

$$V^{\pi}(s) = E_{\pi}\left\{R_t \middle| s_t = s\right\} = E_{\pi}\left\{r_{t+1} + \gamma V(s_{t+1}) \middle| s_t = s\right\} = \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^{\pi}(s')\right] \tag{3}$$

首先通过（2）式构造一个返回函数 R_t，用于反映系统在某个策略 π 指导下的一次学习循环中，从 s_t 状态往后所获得的所有奖赏的累计折扣和。由于环境是不确定的，系统在某个策略 π 指导下的每一次学习循环中所得到的 R_t 有可能是不同的。因此在 s 状态下的值函数要考虑不同学习循环中所有返回函数的数学期望。因此在 π 策略下，系统在 s 状态下的值函数由（3）式定义，其反映了如果系统遵循 π 策略，所能获得的期望的累计奖赏折扣和。

根据 Bellman 最优策略公式，在最优策略 π^*下，系统在 s 状态下的值函数由（4）式定义。

$$V^*(s) = \max_{a \in A(s)} E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right\} = \max_{a \in A(s)} \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^*(s')\right] \tag{4}$$

在动态规划技术中，在已知状态转移概率函数 P 和奖赏函数 R 的环境模型知识前提下，从任意设定的策略 π_0 出发，可以采用策略迭代的方法（式 5 和式 6）逼近最优的 V^*和 π^*，如图 2 中的虚线所示。式 5 和式 6 中的 k 为迭代步数。

$$\pi_k(s) = \arg\max_a \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^{\pi_{k-1}}(s')\right] \tag{5}$$

$$V^{\pi_k}(s) \leftarrow \sum_a \pi_{k-1}(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^{\pi_{k-1}}(s')\right] \tag{6}$$

但由于强化学习中，P 函数和 R 函数未知，系统无法直接通过（5）式、（6）式进行值函数计算。因而实际中常采用逼近的方法进行值函数的估计，其中最主要的方法之一是 Monte Carlo 采样，如（7）式。其中 R_t 是指当系统采用某种策略 π，从 s_t 状态出发获得的真实的累计

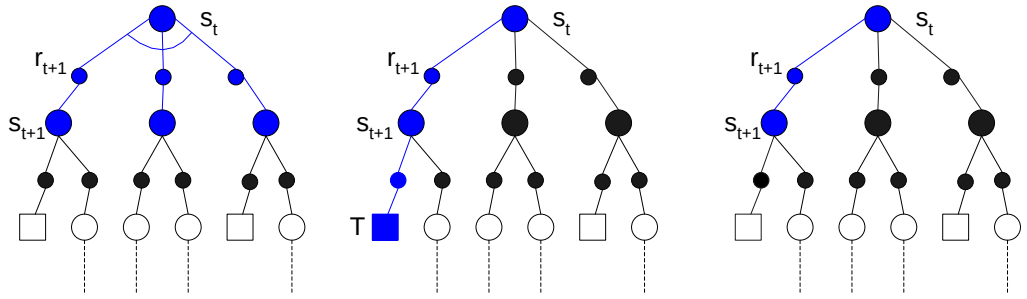
折扣奖赏值。保持 π 策略不变，在每次学习循环中重复地使用（7）式，下式将逼近（3）式。

$$V(s_t) \leftarrow V(s_t) + \alpha\left[R_t - V(s_t)\right] \tag{7}$$

结合 Monte Carlo 方法和动态规划技术，式（8）给出强化学习中时间差分学习（TD，Temporal difference）的值函数迭代公式。

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right] \tag{8}$$

图 3 比较了动态规划技术、Monte Carlo 方法和强化学习方法计算状态值函数的异同。图 3（a）中，动态规划方法考察环境的概率模型，从而用（6）式 Bootstrapping 方法计算所有可能分支所获得奖惩返回值的加权和；图 3（b）中，Monte Carlo 方法采样一次学习循环所获得的奖惩返回值。然后通过多次学习，用实际获得的奖惩返回值去逼近真实的状态值函数。逼近公式即前述的（7）式；图 3（c）中，TD 方法和 Monte Carlo 方法类似，仍然采样一次学习循环中获得的瞬时奖惩反馈，但同时类似与动态规划方法采用 Bootstrapping 方法估计状态的值函数。然后通过多次迭代学习，采用（8）式去逼近真实的状态值函数。图 3 中蓝线表示每种方法计算值函数时所需要的信息。



（a）动态规划方法　　（b）Monte Carlo 方法　　（c）TD 方法

图 3 三种不同的计算值函数方法

表 1 给出强化学习技术中经典的 TD(0)学习算法。表 1 学习算法事实上包含了两个步骤，

表 1. TD(0)算法

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode)
    Choose a from s using policy π derived from V(e.g., ε -greedy)
    Take action a, observer r, s'
    $V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$
    $s \leftarrow s'$
  Until s is terminal

其一是从当前学习循环的值函数确定新的行为策略；其二是在新的行为策略指导下，通过所获得的瞬时奖惩值对该策略进行评估。学习循环过程如图 4 所示，直到值函数和策略收敛。

$$v_0 \to \pi_1 \to v_1 \to \pi_2 \to \cdots \to v^* \to \pi^* \to v^*$$

图 4 强化学习策略迭代过程

再回到（7）式和（8）式上，Monte Carlo 方法采用一次学习循环所获得的整个返回函数去逼近实际的值函数，而强化学习方法使用下一状态的值函数（即 Bootstrapping 方法）和当前获得的瞬时奖赏来估计当前状态值函数。显然，强化学习方法将需要更多次学习循环才能逼近实际的值函数。因此，给我们带来的思考是——能否在（8）式值函数更新中，不仅仅依赖当前状态的瞬时奖赏值，也可以利用下一状态的瞬时奖赏值，一直到终结状态？如果可以，那么如何修改（8）式呢？同时如何修改学习算法呢？

为回答这个问题，构造一个新的 λ-返回函数 R_t'，如式（9），其中假定系统在此次学习循环中第 T 步后进入终结状态。λ-返回函数 R_t'的物理意义如图 5 所示。那么值函数迭代即遵循式（10）。但由于强化学习算法中值函数的更新是在每一学习步（即每次获得<*s, a, r, s'*>经验后）进行的，因此为使学习算法能在一次学习循环中值函数满足（10）式，表 2 设计新的 TD(λ)算法。在表 2 中通过构造 *e*(*s*)函数，即可以保证在一次学习循环中值函数以（10）式更新。

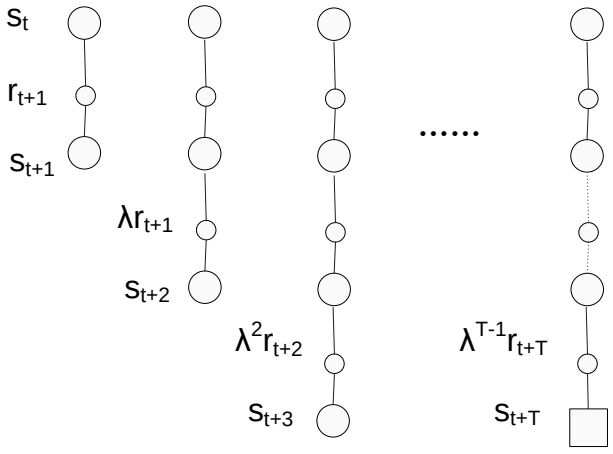


图 5 λ-返回函数

$$R_t' = r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \cdots + \lambda^{T-1} r_{t+T} \qquad (9)$$

$$V(s_t) \leftarrow V(s_t) + \alpha\left[R_t' - V(s_t)\right] \qquad (10)$$

在表 1 和表 2 的策略迭代过程中，我们可以将值函数的估计和策略评估两步骤合二为一。在算法中构造状态-动作对值函数，即 Q 函数。Q 函数定义如下式。理论证明，当学习率 α 满足一定条件[2]，Q 学习算法必然收敛于最优状态-动作对值函数[TJ94]。Q 学习算法是目前最普遍使用的强化学习算法之一。

$$Q^{\pi}(s,a) = \sum_{s'} P_{ss'}^{a}\left[R_{ss'}^{a} + \gamma V^{\pi}(s')\right] \qquad (11)$$

[2] $\sum_{t=1}^{\infty}\alpha_t = \infty$， $\sum_{t=1}^{\infty}\alpha_t^2 < C < \infty$。C 为常数。

表 2. TD(λ)算法

Initialize $V(s)$ arbitrarily and $e(s)=0$ for all $s \in S$
Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode)
    $a \leftarrow$ action given by π for s(e.g., ε -greedy)
    Take action a, observer r, s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    for all s
      $V(s) \leftarrow V(s) + \alpha \delta e(s)$
      $e(s) \leftarrow \gamma \lambda e(s)$
    s $\leftarrow s'$
  Until s is terminal

表 3. Q 学习算法

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode)
    Choose a from s using policy derived from Q (e.g., ε -greedy)
    Take action a, observer r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$
  Until s is terminal

3 部分感知马氏决策过程中的强化学习

在实际的问题中，系统往往无法完全感知环境状态信息。即使环境属于马尔可夫型，但由于感知的不全面，对于状态之间的差异也无法区别。因此部分感知问题属于非马尔可夫型环境[Lov91]。在部分感知问题中，如果不对强化学习算法进行任何处理就加以应用的话，学习算法将无法收敛。

在部分感知模型中，不仅考虑动作的不确定性，同时也考虑状态的不确定性。这种环境描述更接近现实世界，因此应用面比马氏决策模型更广。解决部分感知问题的基本思路是将部分感知环境转换为马氏决策模型描述，即假设存在部分可观测（或不可观测）的隐状态集 S 满足马尔可夫属性。

下面首先给出 POMDP（Partially Observable Markov Decision Process）模型的定义[KLC98]。

POMDP 由六元组<*S, A, R, P,Ω,O*>定义。其中<*S, A, P, R*>定义了环境潜在的马尔可夫决策模型上，*Ω* 是观察的集合，即系统可以感知的世界状态集合[3]，观察函数 *O*：*S*×*A*→PD（*Ω*）。系统在采取动作 *a* 转移到状态 *s′*时，观察函数 *O* 确定其在可能观察上的概率分布。记为 *O*（*s′, a, o*）。

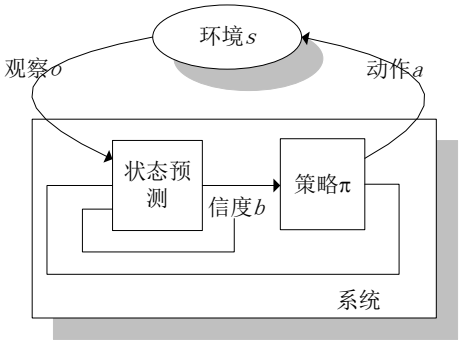


图 6　POMDP 问题结构

解决 POMDP 问题目前最主要的研究方法是预测模型法，如图 6。在预测模型方法中，将状态迁移的历史知识应用于预测模型或构建系统的内部状态，同时引入对内部状态的置信度，将 POMDP 问题转化为统计上的 MDP 求解[KLC98]。状态的置信度又称为信度状态（Belief state）。信度状态 *b* 定义为在隐状态集 *S* 上的概率分布，记 *b*(*s*)为对状态 *s* 的置信度。在信度状态 *b*，系统执行动作 *a*，得到新的观察 *o*，此时根据 Bayes 原理，新信度状态 *b′*计算如下：

$$b'(s') = \Pr(s'|o,a,b) = \frac{O(s',a,o)\sum_{s\in S} P(s,a,s')b(s)}{\Pr(o|a,b)} \qquad (12)$$

上式意义在于：给定一个隐状态集合 *S* 上的概率分布 *b*(*s*)，系统执行 *a* 动作，系统转移到状态 *s′*的概率由分子求和项部分计算。但在新观察 *o* 的约束下，分子乘上观察函数以确定在 *s′* 状态的置信度。分母事实上是归一化项。显然，当信度状态是可计算时，POMDP 问题最优策略学习转变为“信度状态 MDP”（Belief MDP）最优策略的学习。信度状态 MDP 模型定义如下：

Belief MDP *B* 是系统所有信度状态的集合，*A* 是动作集合，记状态转移函数为 *τ*（*b*，*a*，*b′*），奖赏函数为 *ρ*（*b*，*a*）。显然，

$$\tau(b,a,b') = \sum_{o\in O} \Pr(b'|a,b,o)\Pr(o|a,b) \qquad (13)$$

$$\rho(b,a) = \sum_{s\in S} b(s)R(s,a) \qquad (14)$$

根据 Belief MDP 定义，结合 Q 学习，L. P. Kaelbling 等人给出求解 POMDP 精确解算法[KLC98]，其学习过程如图 7 所示。但由于信度状态 MDP 模型是一个连续状态的模型，随着环境复杂程度增加（|*S*|＞15，|*O*|＞15），预测模型的大小呈爆炸性的增大，算法实际上不可行。因此如何结合下节的函数估计方法有效地减少计算量、加快学习算法的收敛速度是有待解决的研究课题之一。

[3] *Ω*可以是 *S* 的子集，也可以与 *S* 无关

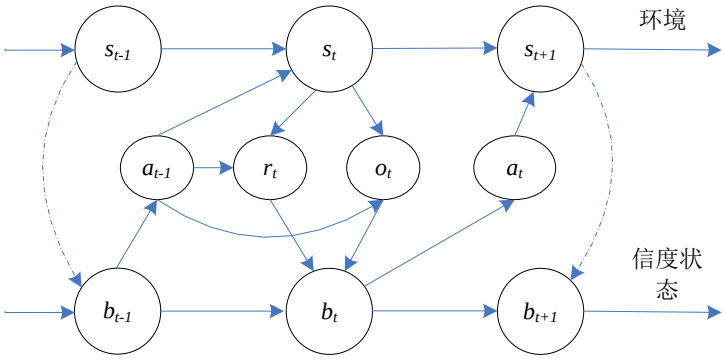


图 7　Belief MDP 学习过程

对 POMDP 问题的学习，目前是强化学习中一个非常重要的研究方向。除了精确求解 POMDP 问题最优策略的强化学习算法之外，研究人员还提出了一些逼近算法。这些算法基本上可以分为两大类，一类是逼近最优值函数，另一类是直接逼近最优策略[Sam02]。主要算法和基本思想列于表 4 中。目前来看，采用一些启发式方法（包括第 5 节的分层方法）解决 POMDP 问题，仍是研究的重点之一。

表 4　强化学习解决 POMDP 问题主要方法

	算法名称	基本思想
学习值函数	Memoryless policies	直接采用标准的强化学习算法
	Simple memory based approaches	使用 k 个历史观察表示当前状态
	UDM(Utile Distinction Memory)	分解状态，构建有限状态机模型
	NSM(Nearest Sequence Memory)	存储状态历史，进行距离度量
	USM(Utile Suffix Memory)	综合 UDM 和 NSM 两种方法
	Recurrent-Q	使用循环神经网络进行状态预测
策略搜索	Evolutionary algorithms	使用遗传算法直接进行策略搜索
	Gradient ascent method	使用梯度下降（上升）法搜索

4 强化学习中的函数估计

对于大规模 MDP 或连续空间 MDP 问题中，强化学习不可能遍历所有状态。因此要求强化学习的值函数具有一定泛化能力。强化学习中的映射关系包括：$S\to A$、$S\to R$、$S\times A\to R$、$S\times A\to S$ 等等。强化学习中的函数估计本质就是用参数化的函数逼近这些映射。

用算子 Γ 来表示本章第 2 节中的（8）式。假设初始的值函数记为 V_0，则学习过程产生的值函数逼近序列为：

$$V_0,\Gamma(V_0),\Gamma(\Gamma(V_0)),\Gamma(\Gamma(\Gamma(V_0))),\ldots\ldots$$

在经典的强化学习算法中，值函数采用查找表（lookup-table）方式保存。而在函数估计

中，采用参数化的拟合函数替代查找表。此时，强化学习基本结构如图 8 所示。记函数估计中V 为目标函数，V'为估计函数，则 $M: V\rightarrow V'$为函数估计算子。假设值函数初值为 V_0，则学习过程中产生的值函数序列为：

$$V_0, M(V_0), \Gamma(M(V_0)), M(\Gamma(M(V_0))), \Gamma(M(\Gamma(M(V_0)))),\ldots\ldots$$

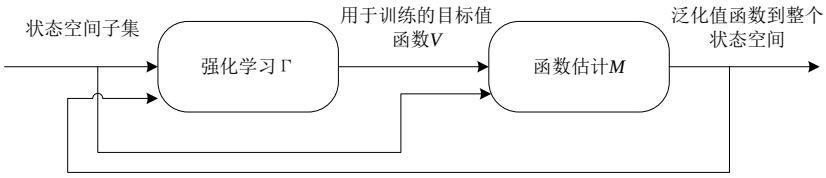


图 8　有函数估计的强化学习结构

因此，类似与 Q 学习算法，使用函数估计的强化学习算法迭代公式做以下修改。

$$Q(s,a)\leftarrow(1-\alpha)V'(s,a)+\alpha\left(r(s,a,s')+\max_{a'}V'(s',a')\right) \quad (15)$$

$$V'(s,a)=M(Q(s,a)) \quad (16)$$

在函数估计强化学习中，同时并行两个迭代过程：一是值函数迭代过程 Γ，另一是值函数逼近过程 M。因此 M 过程逼近的正确性和速度都将对强化学习产生根本的影响。目前函数估计的方法通常采用有导师监督学习方法：如状态聚类[SJJ95][Moo94]、函数插值[Dav97]和人工神经网络[Sut96]等方法。

状态聚类将整个状态空间分成若干区域，在同一区域的状态认为其值函数相等。于是一个连续或较大规模的 MDP 问题被离散化为规模较小的 MDP 问题。状态聚类最简单的方法是区格法，它将状态空间的每一维等分为若干区间，而将整个状态空间划分为若干相同大小的区间，对二维来说就是区格划分。更复杂的划分方法是变步长划分和三角划分，采用状态聚类方法的函数估计强化学习已经被证明是收敛的。需要指明的是：尽管状态聚类强化学习是收敛的，但其并不一定收敛到原问题的最优解上。要使收敛的值函数达到一定的精度，状态聚类的步长不能太大。因此对于大规模 MDP 问题，它仍然面临着维数灾难的困难。

线性插值和多线性插值是状态聚类的改进，它并不将一个区间（或区格）的值函数设为一个值，而是对顶点进行线性插值，从而可以取得更好的性能。Davies 等研究在一个二维的问题上，使用 11*11=121 的区格上的双线性插值便可以取得 301*301=90601 的区格法相当的性能[Dav97]。线性插值和多线性插值也已被证明是收敛的，但其仍然面临“维数灾难”的困难。而非线性插值则不能保证收敛性[Gor95]。

目前函数估计强化学习研究的热点是采用神经网络等方法进行函数估计。但尽管这些新方法可以大幅度提高强化学习的学习速度，但并不能够保证收敛性。因此研究既能保证收敛性，又能提高收敛速度的新型函数估计方法，仍然是目前函数估计强化学习研究的重点之一。构造一个 V 的估计函数$V(s)=\Im(s,\vec{\theta})$，其中 $\vec{\theta}$ 是估计函数的参数。表 5 给出对估计函数进行梯度下降法逼近的强化学习算法框架，其中$\nabla_{\vec{\theta}}$是对估计函数中的参数求导。

表 5. 使用梯度下降法进行函数估计的 TD(λ)算法

Initialize $V(s)$, $\vec{\theta}$ arbitrarily and $\vec{e}(s)=\vec{0}$ =0 for all $s\in S$

Repeat (for each episode)

Initialize s

Repeat (for each step of episode)

$a\leftarrow$ action given by π for s(e.g., ε -greedy)

Take action a, observer r, s'

$\delta\leftarrow r+\gamma V(s')-V(s)$

$\vec{e}\leftarrow\gamma\lambda\vec{e}+\nabla_{\vec{\theta}}V(s)$

$\vec{\theta}\leftarrow\vec{\theta}+\alpha\delta\vec{e}$

$s\leftarrow s'$

Until s is terminal

5 分层强化学习

经典马氏决策过程模型只考虑了决策的顺序性而忽略决策的时间性。基于马氏决策过程的强化学习都假设动作在单个时间步完成，因而无法处理需要在多个时间步完成的动作。为解决此问题，引入半马氏决策过程（SMDP，Semi-MDP）模型。在 SMDP 模型中，每个行为动作的时间间隔作为变量（整数或实数），并进一步可以细分为连续时间-离散事件 SMDP 和离散时间 SMDP 两种模型。在后者中，行为决策只在单位时间片的正整数倍做出，较前者模型简单。但基于离散时间 SMDP 的强化学习方法不难推广到连续时间的情况，因此以下的讨论都基于离散时间 SMDP。

SMDP 记 τ 是系统在状态 s 执行动作 a 后的随机等待时间，$P(s', \tau \mid s, a)$ 是 τ 时间步后执行动作 a 从状态 s 转移到状态 s'的状态转移函数，$R(s, a)=E\{r_t+\gamma r_{t+1}+\ldots+\gamma^{\tau}r_{t+\tau}\}$为对应的奖赏值[BM03]。

基于 SMDP 的值函数 Bellman 最优公式、状态-动作对值函数 Bellman 最优公式以及 Q 学习迭代公式分别如以下三式，其中 k 为迭代次数。

$$V^*(s)=\max_{a\in A}\left[R(s,a)+\sum_{s',\tau}\gamma^{\tau}P(s',\tau\mid s,a)V^*(s')\right] \tag{17}$$

$$Q^*(s,a)=R(s,a)+\sum_{s',\tau}\gamma^{\tau}P(s',\tau\mid s,a)\max_{a'\in A}Q^*(s',a') \tag{18}$$

$$Q_{k+1}(s,a)\leftarrow(1-\alpha)Q_k(s,a)+\alpha\left[r_t+\gamma r_{t+1}+\cdots+\gamma^{\tau-1}r_{t+\tau-1}+\gamma^{\tau}\max_{a'\in A}Q_k(s',a')\right] \tag{19}$$

为解决大规模强化学习中的维数灾难问题，可以通过抽象来简化问题的描述。最主要抽象的方法是建立宏动作（Macro），每个宏动作包含一个动作系列，可被系统或其他宏直接调用，从而形成了分层强化学习的控制机制。此时，分层强化学习不必在每个时间步都做出决

策，因此可以进一步用前面的半马氏决策过程进行建模。目前分层强化学习中三种典型的方法有 Option[SPS99]、HAM[PR97]和 MAXQ[Die00]。由于篇幅关系，下面我们主要讨论 Option 算法的基本原理。需要特别指出的是，MAXQ 以及其发展 HEXQ 目前正逐步成为分层强化学习的研究热点。

Option（抉择）是对 MDP 中元动作的扩展，它在原来元动作的概念中加入了动作执行过程这一指标。下面首先给出 Option 的定义。

Option 每个 option 由一个三元组<*I*, *π*, *β*>表示。其中，$\pi : S \times \bigcup_{s \in S} A_s \to [0,1]$ 表示单个 option 的内部策略，$\beta : S \to [0,1]$ 表示终结状态判断条件，$I : I \subseteq S$ 为激活 option 时的初始状态[SPS99]。

一个 option 被激活当且仅当当前状态属于初始状态集 *I*，option 在执行时动作的选择依赖于内部策略 *π*，最终当系统判断 option 当前状态为某一终结状态时，整个 option 执行结束。举例来说，如果 option 的当前状态为 *s*，根据内部策略 *π*（*s*, *a*）选择某一动作，当确定下一动作 *a* 后动作作用于环境使环境状态转移为 *s′*，然后根据 *β*（*s′*）所给出的概率决定 option 是终结还是继续执行。若当前 option 执行结束，系统可以选择另一个 option 开始执行。

我们通常假设系统在任何状态都可以选择某一 option 开始执行，因此可以认为每个元动作也是一个 option。于是 option 集合中包含两种类型的 option：一种是在单时间步完成；另一种则需要执行若干时间步才能完成。

经典马氏决策模型中的值函数概念可以被很自然的运用到 option 上。我们定义 $Q^{\mu}(s, o)$为当前状态 *s* 下，系统依据策略 *μ*，选择执行 option *o* 所得的值函数如下：

$$Q^{\mu}(s,o) = E\left\{ r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{\tau-1} r_{t+\tau} + \cdots \mid \varepsilon(o\mu, s, t) \right\} \tag{20}$$

这里 *oμ* 表示系统在 option *o* 内部策略执行 *τ* 步使得 *o* 终结后，再由策略 *μ* 决定下一状态。*ε*（*oμ*，*s*，*t*）表示系统在 *t* 时刻的执行过程。

通过以上分析，我们很容易看出 Option 算法与 SMDP 的关系。这里 SMDP 可以理解为在有穷状态 MDP 中加入一组确定的 options，这时系统的动作只在 options 集合内选择，并且只有在系统执行完整个 option 后环境才返还给奖赏值[SPS99]。这样我们就可以运用 SMDP 的学习算法来解决在给定的 option 集合上寻找最优策略的问题。

我们利用单步 Q 学习算法来对值函数进行学习，值函数的每次更新都发生在 option 执行结束之后。假设 option *o* 在状态 *s* 开始执行，且在状态 *s′*终结，函数 $Q(s, o)$的迭代算法如式 21。这里的 option *o* 在执行了 *τ* 步后在状态 *s′*处终结，*r* 为 option *o* 的在整个执行过程中的累计折扣奖赏值。与传统 Q 学习相同的收敛条件下，$Q(s, o)$总是收敛于最优状态-动作对值函数。

$$Q_{k+1}(s,o) \leftarrow (1-\alpha) Q_k(s,o) + \alpha \left[r + \gamma^{\tau} \max_{o' \in O_{s'}} Q_k(s', o') \right] \tag{21}$$

在分层强化学习环境下，运用 Option 算法已经被证明不仅可以大大加速当前任务的学习速度，而且已学得的知识还可以应用在今后碰到的其它类似的学习任务之中[Ber99][Iba89][Pre00]。目前，构造 option 最简单的方法就是随机或是启发式的产生若干 options，然后由系统对该组 options 测试来进行筛选。虽然这样得到的 options 有可能大大提高系统的决

策效率，但是筛选的过程往往也对系统的性能负面影响。这就要求我们尝试让系统能利用通过学习已经得到的策略，自主的在整个状态空间中找到一组合适的子目标，进而构造相应的一组 options。目前自主寻找子目标，通常由子目标的“瓶颈”特性入手，通过记录学习过程中访问频率最高的若干状态作为子目标[Dig98]。当获得问题的子目标后，我们假设状态空间的每个子空间都各自对应于一个 option，于是构造 options 的过程可以按如下方式进行：把该子空间的子目标作为相应 option 的终结状态，把子空间内除子目标以外的所有的状态的集合作为起始状态集 I，option 的内部策略 π 可以在起始状态集和终结条件确定后利用强化学习算法得到。文献[苏高 05]给出了强化学习自主寻找子目标并根据子目标构造 options 的算法框架。

6 多 agent 强化学习

多 agent 强化学习是强化学习研究中非常重要的研究方向之一。在多 agent 系统中，环境在多个 agent 的联合动作下进行状态的迁移。对于单个 agent 来讲，由于其只能确定自身 agent 的行为动作，因此体现出一种行为动作上的“部分感知”，从而产生出另一种形式的非标准马尔可夫环境。多 agent 强化学习机制被广泛应用到各个领域，例如游戏、邮件路由选择、电梯群控系统以及机器人设计等等。

Weiss 将多 agent 学习分成三类：乘积（multiplication）形式、分割（division）形式和交互（interaction）形式[WD98]。这种分类方法要么将多 agent 系统作为一个可计算的学习 agent；要么是每个 agent 都有独立的强化学习机制，通过与其他 agent 适当交互加快学习过程。每个 agent 拥有独立的学习机制，并不与其他 agent 交互的强化学习算法称之为 CIRL（Concurrent Isolated RL）。CIRL 算法只能够应用在合作多 Agent 系统，并只在某些环境中优于单 agent 强化学习。而每个 agent 拥有独立的学习机制，并与其他 agent 交互的强化学习算法称之为交互强化学习（Interactive RL）。不同与单 agent 强化学习只考虑时间信用分配问题（Temporal credit assignment problem），交互式强化学习面临的主要问题是结构信用分配问题（Structural credit assignment problem），即整个系统获得的奖赏如何分配到每个 agent 的行为上，典型算法包括 ACE 和 AGE 等[Ger99]。

但 Weiss 关于多 agent 强化学习的讨论并不能覆盖当前多 agent 强化学习大部分研究内容。事实上，在多 agent 强化学习中存在两种角度的研究方法。一是从机器学习角度着手；另一种是从多 agent 系统角度分析。在表 6 中，我们给出三种主要的多 agent 强化学习技术，然后再进行具体分析。

对于第一种类型的多 agent 强化学习，我们称之为合作多 agent 强化学习（Cooperative multiagent reinforcement learning）。这种多 agent 强化学习更多地是强调如何利用分布式强化学习来提高强化学习的学习速度。早在九十年代初，Tan 等就指出合作多 agent 强化学习中相互交互（交换信息）是最有效的方法之一[Tan93]。Tan 给出三种主要的实现方法：1）交换每个 agent 感知的状态信息；2）交换 agent 学习的经验片段（即<*s, a, r, s'*>经验）；3）交换学习过程中的策略或参数等。Luis Nunes 等在 2004 年又给出第四种方法：4）交换建议[LE03]。所有这些方法相比与单 agent 强化学习，都能够有效提高学习速度。我们认为合作多 agent 强化学习的基本思想在于“**在 agent 进行动作选择前，相互交互，产生更新后的值函数，而动作的选**

择基于新的值函数。”但是，Luis Nunes 等研究人员明确指出合作多 agent 强化学习的主要问题在于“*When, why, how to exchange information* ?”，“*A thorough analysis of the conditions in which this technique is advantageous is still necessary*.”

表 6. 三种主要多 agent 强化学习技术

	问题空间	主要方法	算法准则
合作多 agent 强化学习	分布、同构、合作环境	交换状态 交换经验 交换策略 交换建议	提高学习收敛速度
基于平衡解多 agent 强化学习	同构或异构、合作或竞争环境	极小极大-Q NASH-Q CE-Q WoLF	理性和收敛性
最佳响应多 agent 强化学习	异构、竞争环境	PHC IGA GIGA GIGA-WoLF	收敛性和不遗憾性

为了回答这个问题，并理论分析多 agent 强化学习中的交互作用，研究者借助于对策论（博弈论，game theory）数学工具对多 agent 强化学习进行进一步分析。在对策模型中，每个 agent 获得的瞬时奖惩不仅仅取决于自身的动作，同时还依赖于其他 agent 的动作。因此可以将多 agent 强化学习系统中每个离散状态 s 形式化为一个对策 g。那么强化学习的马尔可夫决策模型扩展为多 agent 系统的马尔可夫对策模型。我们称这种形式的多 agent 强化学习为基于对策多 agent 强化学习（Game-based multiagent reinforcement learning）或基于平衡解多 agent 强化学习(Equilibrium-based multiagent reinforcement learning)。

下面首先给出马尔可夫对策模型的形式定义。

马尔可夫对策 在 n 个 agent 的系统中，定义离散的状态集 S（即对策集合 G），agent 动作集 A_i 的集合 A，联合奖赏函数 R_i：$S\times A_1\times\ldots\times A_n\rightarrow\mathcal{R}$ 和状态转移函数 P：$S\times A_1\times\ldots\times A_n\rightarrow$PD（$S$）。每个 agent 目标都是最大化期望折扣奖赏和。

Agent B \ Agent A	剪刀	石头	布
剪刀	(0,0)	(-1,+1)	(+1,-1)
石头	(+1,-1)	(0,0)	(-1,+1)
布	(-1,+1)	(+1,-1)	(0,0)

（a）剪刀-石头-布

Agent B \ Agent A	对抗	合作
对抗	(-9,-9)	(0,-10)
合作	(-10,0)	(-1,-1)

（b）囚犯两难

图 9 零和及广义和对策模型

多 agent 对策模型基本上可以分为零和对策和广义和对策两种。前者中，强调在系统的任何状态下，多个 agent 从环境中获得的奖赏总和为 0；而在广义和对策中，则没有此要求。下图以两 agent 为例，图 9（a）给出“剪刀-石头-布”游戏的零和对策模型，图 9（b）给出“囚犯两难”的广义和对策模型。

由于 agent A 的奖赏值取决于 agent B 的动作，因此传统单 agent 强化学习算法在零和对策的强化学习中不适用。解决这一问题最简单的方法是采用极小极大 Q 算法：在每个状态 s，对于 agent A 其最优策略为 agent B 选择最坏动作情况下，agent A 选择奖赏最大的动作[Lit94]。因此定义零和对策多 agent 强化学习的值函数为：

$$V(s)=\max_{a\in A}\min_{\vec{b}\in\vec{A}}\left(Q\left(s,a,\vec{b}\right)\right) \tag{20}$$

显然如果将马尔可夫对策中每个状态都形式化为如图 9（a）的零和对策模型，那么极小极大 Q 算法可以发现最优的策略。但对于图 9（b）的非零和对策模型，如果采用极小极大算法求解，其最优解为（对抗，对抗），奖赏为（-9，-9）；而显然囚犯两难问题的最优解为（合作，合作），奖赏为（-1，-1）。因此在非零和马尔可夫对策模型中，用极小极大 Q 算法得不到最优解。其原因在于非零和对策模型更能反应多 agent 系统中个体理性（individual rationality）与集体理性（group rationality）冲突的本质。

Hu 等借助于对策理论中的 Nash 均衡解概念，设计 Nash-Q 学习算法求解非零和马尔可夫对策中的最优策略[HW98]。基于 Hu 的方法，Littman 等同时考虑零和和非零和两种对策模型，提出 Friend-and-Foe-Q 学习算法[Lit01]；而 Greenwald 等引进相关均衡解，设计 CE-Q 算法来综合 Nash-Q 学习算法和 Friend-and-Foe-Q 学习算法[GHS03]。在所有基于平衡解的多 agent 强化学习算法中，算法必须满足两个性质——“理性”和“收敛性”。前者说明当其他 agent 采用固定策略时，基于平衡解的强化学习算法应能够收敛到“最优反应策略”（Best-response policy）；而后者强调当所有 agent 都采用基于平衡解的强化学习算法时，算法必然收敛到稳定策略，而不能出现振荡现象。同样我们认为：所有基于平衡解多 agent 强化学习的基本思想在于“**在 agent 进行动作选择时，选择动作不再仅仅依赖自身的值函数，而必须同时考虑其他 agent 的值函数，选择的动作是在当前所有 agent 值函数下的某种平衡解**”。

不同于基于平衡解的多 agent 强化学习算法，最佳响应多 agent 强化学习（Best-response multiagent reinforcement learning）着重研究无论其他 agent 采用何种策略，算法如何获得最优策略。算法设计的两个主要准则是收敛准则和不遗憾准则。收敛准则与基于平衡解的多 agent 强化学习算法是一致的。而不遗憾准则是指：当对手策略稳定的情况下，采用最佳响应学习算法策略的 agent 所获得的奖赏要大于等于采用任意纯策略所获得的奖赏。

最佳响应多 agent 强化学习的基本思路是对对手策略进行建模或者是对自身策略进行优化。目前有三种主要的方法：PHC（Policy Hill Climbing）算法根据自身的策略历史进行策略的更新，以最大化获得的奖赏[CK01]；第二种方法是根据对对手策略的观察，来优化自身策略[CB98]；而第三种方法通过梯度上升法来调整每个 agent 的策略，以增大其期望奖赏值。典型的算法有 IGA 及其变种 GIGA、WoLF 和 GIGA-WoLF 等[Bow04]。我们认为：最佳响应多 agent 强化学习的基本思想在于“**在 agent 进行动作选择时，选择动作不仅仅依赖自身的值函数，而必须同时考虑自身历史的策略和其他 agent 策略的估计**”。

在目前多 agent 强化学习研究中，三种类型的多 agent 强化学习是独立发展的。目前研究中，一方面将继续发展满足各自不同应用需求的算法（如加速收敛、或理性、或不遗憾性等）；另一方面，也将研究不同类型多 agent 强化学习算法之间的关系，希望能够形成一种统一的框架。

8 结束语

本章综述了强化学习技术基本原理和目前主要研究方向。尽管在过去的二十年中，强化学习技术研究取得了突破性进展，但目前仍然存在许多有待解决的问题。我们认为，在今后的若干年中，除了本章所讨论的研究方向外，以下方面也将成为强化学习研究的重要研究内容。

强化学习与其他学习技术相结合的研究。众所周知，强化学习的一个主要缺点是收敛慢。其根本原因在于学习过程仅仅从经验获得的奖赏中进行策略的改进，而忽略了大量其他有用的领域信息。因此，如何结合其他机器学习技术，如神经网络、符号学习等技术，来帮助系统加快学习速度是强化学习研究和应用的重要方向。目前，结合技术研究的主要难点在于：如何从理论上证明和保证学习算法的收敛性。

非马氏决策过程中的新型强化学习算法研究。经典的马氏决策模型是相当简单的，除了本章讨论的部分感知、连续状态、半马氏决策过程等模型外，在实际应用中还存在大量更加复杂的模型。例如，在图象的马尔可夫随机场模型中，状态的迁移是由历史多个相邻状态决定。因此，在更复杂马氏决策模型中发展有效的强化学习算法也将是未来重要的研究方向之一。

强化学习应用研究。目前，强化学习的应用主要可以分为四类：制造过程控制、各种任务调度、机器人设计和游戏。另外，强化学习在学习分类器（Learning Classifier System）中的应用也逐渐成为研究的热点。从当前看来，强化学习的应用逐步向一些新的机器学习任务上拓展，如 Web Log Mining、Web Crawling、Classification 等等。因此，如何在新应用上快速、有效地部署和应用强化学习技术也是放在研究人员面前的挑战之一。

但是我们相信，在研究者的努力和实际应用的需求的带动下，强化学习技术必将取得更大的发展、发挥更大的作用。

参 考 文 献

[Ber99] Bernstein D S. Reusing old policies to accelerate learning on new MDPs. Technical Report : UM-CS-1999- 026, Dept. of CS, U. of Massachusetts, Amherst, MA, 1999.

[BM03] A G Barto, S Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamics Systems: Theory and Applications*, 2003, 13(4):41-77.

[Bow04] M Bowling. Convergence and no-regret in multiagent learning. In: Advances in Naural Information Processing Systems, 2004.

[CB98] C Claus, C Boutilier. The dynamics of reinforcement learning in cooperative multiagent system. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial

Intelligence/Innovative Applications on Artificial Intelligence, Madison, Wisconsin, United States: American Association for Artificial Intelligence, 1998, 746-752.

[CK01] Y Chang, L Kaelbling. Playing is believing: the role of beliefs in multi-agent learning. In: Proceedings of NIPS-2001, Vancouver, Canada, 2001.

[Dav97] S Davies. Multidimensional triangulation and interpolation for reinforcement learning. In: Michael C Mozer, Michael I Jordan, Thomas Petsche, eds. Advances in Neural Information Processing Systems 9, NY: MIT Press, 1997, 1005-1010.

[Die00] Dietterich T G. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 2000, 13, 227-303.

[Dig98] Digney B. Learning hierarchical control structure for multiple tasks and changing environments. In: Proceedings of the Fifth Conference on the Simulation of Adaptive Behavior: SAB 98, 1998.

[Ger99] Gerhard Weiss. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.

[GHS03] A Greenwald, K Hall, R Serrano. Correlated-q learning. In: Proceedings of Twentieth International Conference on Machine Learning, Washington DC, 2003, 242-249.

[HW98] J Hu, M P Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 2003, 4:1039-1069.

[Iba89] Iba G A. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 1989, 3:285-317.

[Lit94] M L Littman. Markov games as a framework for multi-agent reinforcement learning. In: Eleventh International Conference on Machine Learning, New Brunswick, 1994, 157-163.

[Lit01] M L Littman. Fierend-or-foe q-learning in general-sum games. In: Proceedings of Eighteenth International Conference on Machine Learning, Williams College: Morgan Kaufman, 2001, 322-328.

[KLM96] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 1996, 4: 237~285.

[KLC98] Kaelbling L P, Littman M L, Cassandra A R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998, 101: 99-134.

[LE03] Luis Nunes, Eugenio Oliveira. Cooperative learning using advice exchange. In: E Alonso et al., eds. Adaptive Agents and Multiagent Systems, Lecture Notes in Computer Science, 2636, Berlin, Heidelberg: Springer-Verlag, 2003, 33-48.

[Lov91] Lovejoy W S. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 1991, 28:47-65.

[Moo94] A W Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. In : Jack D Cowan, Gerald Tesauro, Joshua Alspector, eds. Advances in Neural Information Processing Systems, 6: Morgan Kaufmann Publishers, 1994, 711-718.

[PR97] Parr R, Russell S. Reinforcement learning with hierarchies of machines. In: Proceedings of Advances in Neural Information Processing Systems 10. MIT Press, 1997.

[Pre00] Precup D. Temporal abstraction in reinforcement learning. Doctoral dissertation, U. of Massachusetts, Amherstm, 2000.

[Sam02] Samuel W. Hasinoff. Reinforcement learning for problems with hidden state. Technical Report, University of Toronto, Department of Computer Science, 2002.

[SB98] R S Sutton and A.G. Barto. Reinforcement Learning, Cambridge, MA： MIT Press, 1998.

[SJJ95] S Singh, T Jaakkola, M I Jordan. Reinforcement learning with soft state aggregation. In: G Tesauro, D Touretzky, eds. Advances in Neural Information Processing Systems, 7. Morgan Kaufmann: MIT Press, 1995, 361-368.

[SPS99] Sutton R S, Precup D, Singh, S. Between MDPs and Semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999, 112:181–211.

[Sut96] R S Sutton. Generalization in reinforcement learning: successful examples using sparse coarse coding. In: D Touretzky, M.Mozer, M. Hasselmo, eds. Advances in Neural Information Processing Systems, 8, NY: MIT Press, 1996, 1038-1044.

[Tan93] M Tan. Multi-agent reinforcement learning : independent vs. cooperative agents. In: Proc. Of the tenth international conference on machine learning, Amherst, MA, 1993: 330-337.

[TJ94] Tsitsiklis, John N. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 1994, 16(3):185-202.

[WD98] Weiss G, Dillenbourg P. What is multi in multiagent learning? In: P Dillenbourg, eds. Collaborative learning. Cognitive and computational approaches. Amsterdam: Pergamon Press, 1998, 64-80

[苏高 05] 苏畅, 高阳等. SMDP 环境下自主生成 options 的算法研究. 模式识别与人工智能, 2005.