# 增强学习中的直接策略搜索方法综述

王学宁[1],陈 伟[1],张 锰[2],徐 昕[1],贺汉根[1]
(1.国防科技大学 机电工程与自动化学院,湖南 长沙 410073;2.北京清河大楼 子 9,北京 100085)

**摘 要**:对增强学习中各种策略搜索算法进行了简单介绍,建立了策略梯度方法的理论框架,并且根据这个理论框架的指导,对一些现有的策略梯度算法进行了推广,讨论了近年来出现的提高策略梯度算法收敛速度的几种方法,对于非策略梯度搜索算法的最新进展进行了介绍,对进一步研究工作的方向进行了展望.

**关键词**:增强学习;策略搜索;策略梯度
**中图分类号**:TP242 **文献标识码**:A **文章编号**:1673-4785(2007)01-0016-09

# A survey of direct policy search methods in reinforcement learning

WANG Xue-ning[1],CHEN Wei[1],ZHANG Meng[2],XU Xin[1],HE Han-gen[1]
(1. School of Electromechanical Engineering and Automation, National University of Defense Technology, Changsha 410073, China; 2. Qinghe Building Zi 9, Beijing 100085, China)

**Abstract**: The direct policy search methods in reinforcement learning are described, and the theoretic framework of policy gradient methods is presented. According to this framework, some current policy gradient algorithms are generalized. The new methods of speeding up the policy gradient algorithms are discussed. The new non-policy gradient search methods are also described. Finally, some future directions of research work are also given.

**Keywords**: reinforcement learning; policy search; policy Gradient

增强学习(reinforcement learning,又称为强化学习或再励学习),是近年来兴起的一类机器学习方法.增强学习强调在与环境的交互中学习,学习过程中仅要求获得评价性的反馈信号(reward/reinforcement signal,称为回报或增强信号),以极大化未来的回报为学习目标.增强学习由于不需要给定各种状态下的教师信号,因此对于求解复杂的优化决策问题具有广泛的应用前景[1].目前,增强学习在理论和算法研究方面已取得了许多成果,成为求解序贯(sequential)优化决策问题(通常建模为马氏决策问题,Markov decision problems)的一类有效方法[2-7].

在过去的 10 年中,增强学习的研究主要集中在基于值函数的方法.但是基于值函数的学习方法具有以下几个缺陷:1)基于值函数估计的方法易于寻找确定性的最优策略,但是,许多问题的最优策略往往是随机策略.2)行为值的微小变化可能会引起策略很大的变化,这就使得值函数方法在很多问题中不能保证收敛[8].典型的值函数方法如 Q-学习算法、Sarsa 等方法如果采用函数逼近器,即使在小规模的 MDP 问题中也可能会发散[9-11].3)值函数方法需要找出具有最大值的那个行为,但是如果行为空间是连续的,这将会是一个很难或者很费时的问题.

增强学习的另外一大类方法是直接策略搜索方法.该类方法把策略参数化,并且估算优化指标相对于策略参数的梯度,然后利用这个梯度来调整这些参数,最后得到最优或者局部最优策略.直接策略搜索方法最后得到的策略既可以是确定性策略,也可以是随机性的策略.尽管值函数方法也可以利用 soft-max 方法得到随机策略,但是这需要引进新的参数,并且设定"柔软度"(softness)也比较困难,没有任何理论指导.相对于值函数方法,直接策略搜索方法的收敛性也容易证明.因此,近年来直接策略搜索方法引起了广泛的关注[12-15].

**收稿日期**:2006-07-07.
**基金项目**:国家自然科学基金资助项目(60234030, 60303012).

直接策略搜索算法的核心就是估算优化指标相对于策略参数的梯度,根据在估算这个梯度的过程中是否用到策略相对于自己参数的梯度,可以把直接策略搜索算法分为 2 类:策略梯度方法和非策略梯度方法. 如果用到了策略相对于自己参数的梯度,则属于策略梯度方法,否则属于非策略梯度方法.

由于策略梯度方法的发展历程尚短,因此还没有文章介绍策略梯度方法的理论体系和各种算法. 文中详细地介绍了各种策略梯度算法,并建立了理论框架,在此框架的指导下,对现有的策略梯度方法进行了推广.

## 1　增强学习

增强学习中,往往把问题建模为马氏决策过程(Markov decision process,MDP). Markov 决策过程按照决策时间的特性和模型中其他因素的不同,可以有多种分类方法,如平稳和非平稳 Markov 决策过程、离散时间和连续时间 Markov 决策过程、离散状态空间和连续状态空间 Markov 决策过程等. 在文中将以离散时间平稳 Markov 决策过程为研究对象. 其定义如下:

**定义 1**　离散时间平稳 Markov 决策过程　一个离散时间平稳 Markov 决策过程可以用如下的五元组来表示,即:$\{S, A, r, P, \eta\}$,式中:$S$ 为有限或连续状态空间,$A$ 为有限或连续行为空间,$r: S \times A \to R$ 为回报函数,$P$ 为 MDP 的状态转移概率,满足如下的 Markov 性和时齐特性:

$$\begin{aligned} &\forall i, j \in S, a \in A, \forall n \geq 0, \\ &p(X_{t+1} = j \mid X_t = i, A_t = a, X_{t-1}, A_{t-1}, \cdots, \\ &X_0, A_0) = p(X_{t+1} = j \mid X_t = i, \\ &A_t = a) = p(i, a, j). \end{aligned} \tag{1}$$

式中:$\eta$ 为决策优化的目标函数.

在上述定义中,状态转移概率 $P$ 满足式(2).

$$\sum_j p(i, a, j) = 1. \tag{2}$$

在每个时间点 $t \in \{0, 1, 2, \cdots\}$,状态、行为和回报分别记为 $s_t \in S$, $a_t \in A$, $r_t \in R$.

Markov 决策过程的决策优化目标函数 $\eta$ 主要有 2 种类型. 即折扣总回报目标和平均期望回报目标,分别如式(3)和式(4)所示.

MDP 折扣总回报目标:

$$\eta_d = \mathrm{E}\left[\sum_t^{\infty} \gamma^t r_t\right]. \tag{3}$$

式中:$0 < \gamma < 1$,

MDP 平均期望回报目标:

$$\eta_a = \limsup_{N \to \infty} \frac{1}{N} \mathrm{E}\left[\sum_t^{N-1} r_t\right]. \tag{4}$$

上述 2 种决策优化目标函数在动态规划领域都得到了广泛的研究和应用,在增强学习算法和理论的研究中,主要针对折扣总回报目标函数进行了大量研究. 近年来,针对平均期望回报目标的增强学习方法也取得了一定的研究进展[16-17]. 文献[17]对 3 种目标函数的性能差异进行了深入分析,指出折扣总回报目标函数可以在性能方面近似于平均期望回报目标. 对于有限阶段的 Markov 决策问题,当折扣因子 $\gamma = 1$ 时,2 种目标函数等价. 因此,文中将以具有折扣总回报目标函数的 Markov 决策问题为研究对象.

**定义 2**　Markov 决策过程的一般随机策略　记 $S_t$ 和 $A_t$ 分别为 Markov 决策过程在时刻 $t$ 的状态集和行为集,集合 $\Gamma_t = \{(s, a): s \in S_t, a \in A_t\}$. 则称测度序列 $\pi = (\pi_0, \pi_1, \cdots)$ 为 Markov 决策过程的随机策略,若对于任意 $t \geq 0$,$\pi_t$ 为 $\Gamma_0 \times \Gamma_1 \times \cdots \Gamma_{t-1} \times S_t$ 到 $A_t$ 的转移概率,且满足:

$$\pi_t(A_t(s_t) \mid s_t, a_{t-1}, s_{t-1}, \cdots, a_0, s_0) = 1. \tag{5}$$

定义 2 给出了一般随机策略的严格数学定义,从概念上讲,时刻 $t$ 的策略 $\pi_t$ 确定了在该时刻选择行为的规则. 在理论研究和实际应用中,通常针对一类特殊的策略即马氏策略,其定义如下:

**定义 3**　Markov 决策过程的马氏策略　设 Markov 决策过程的策略 $\pi = (\pi_0, \pi_1, \cdots, \pi_t)$ 满足:

$$\begin{aligned} &\pi_t(a_t(s_t) \mid s_t, a_{t-1}, s_{t-1}, \cdots, a_0, s_0) = \\ &\pi_t(a_t(s_t) \mid s_t) \quad \forall t \geq 0. \end{aligned} \tag{6}$$

则称 $\pi$ 为马氏策略. 若对于任意 $t \geq 1$,有 $\pi_t = \pi_0$,则称马氏策略 $\pi$ 为平稳的.

在定义了 Markov 决策过程的策略后,可以对状态的值函数进行如下定义:

**定义 4**　Markov 决策过程的状态值函数　设 $\pi$ 为平稳策略,则 Markov 决策过程的状态值函数定义为

$$V^{\pi}(s) = \mathrm{E}^{\pi}\left[\sum_t^{\infty} \gamma^t r_t \mid s_0 = s\right]. \tag{7}$$

式中:数学期望 $\mathrm{E}^{\pi}[\ ]$ 定义在状态转移概率 $P$ 和平稳策略 $\pi$ 的分布上.

Markov 决策过程的状态值函数确定了从某一状态出发按照策略 $\pi$ 选择行为所获得的期望总回报的大小. 类似于状态值函数,Markov 决策过程的行为值函数确定了从某一状态-行为对出发,按照策略 $\pi$ 选择行为所获得的期望总回报的大小. 在增强学习算法中,为便于进行策略的更新,通常对行为值

函数进行估计. 下面的定义5给出了Markov决策过程行为值函数的定义:

**定义5** Markov决策过程的行为值函数 设$\pi$为平稳策略,则Markov决策过程的行为值函数定义为

$$Q^{\pi}(s,a) = \mathrm{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r_t \mid s_0 = s, a_0 = a\right]. \quad (8)$$

## 2 策略梯度增强学习及其理论框架

### 2.1 策略梯度方法的理论框架

#### 2.1.1 基本概念和基本理论

策略$\pi$决定了行为的选择,而行为的选择决定了状态的转移概率,状态转移概率直接影响优化指标$\eta$的计算,因此,不同的$\pi$具有不同的$\eta$值,所以优化指标其实是关于策略$\pi$的函数. 直接策略搜索方法的思想就是调整策略的参数,使得优化指标$\eta$达到最大或者局部最大. 称具有需要调整参数的策略为参数化策略.

**定义6** 参数化策略 参数化策略$\pi$是从状态$s \in S$到行为空间上概率分布的映射. 即在给$s \in S$定以及策略的参数$\theta$时,选择行为$a$的概率为$\pi(a|\theta, s)$.

定义了参数化策略之后,自然而然就想到利用梯度$\frac{\partial \eta}{\partial \theta}$来调整参数$\theta$. 一种计算方法为

$$\frac{\partial \eta}{\partial \theta} = \frac{\partial \eta}{\partial \pi}\frac{\partial \pi}{\partial \theta} \quad (9)$$

但是函数关系式$\eta(\pi)$很难求出,尤其是当系统的状态转移概率未知时,无法得到函数$\eta(\pi)$的解析式,因此其梯度也无法利用解析方法计算. 所以,只有寻求其他途径来进行梯度估计. Sutton在文献[8]中提出了一种梯度估计方法:

**定理1** 对于任意给定的MDP,无论是折扣型回报还是平均型回报,都有下式成立

$$\frac{\partial \eta}{\partial \theta} = \sum_{s} d^{\pi}(s) \sum_{a} \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a). \quad (10)$$

式中:$\mathrm{d}^{\pi}(s) = \sum_{t=0}^{\infty}\gamma^{t} P_r\{s_t = s \mid s_0, \pi\}$表示在初始状态为$s_0$的情况下,策略为$\pi$时,达到状态$s$的可能性. 上述梯度估计公式最大的一个优点是:没有出现$\frac{\partial d^{\pi}(s)}{\partial \theta}$,因此,策略的变化对状态的分布影响不大,这就适合用采样方法来估计梯度. 如果状态$s$是采用策略$\pi$时的采样,那么$\sum_{a}\frac{\partial \pi}{\partial \theta}Q^{\pi}(s,a)$就是$\frac{\partial \eta}{\partial \theta}$的无偏估计[8]. 如果采用$\sum_{a}\frac{\partial \pi}{\partial \theta}Q^{\pi}(s,a)$来近似$\frac{\partial \eta}{\partial \theta}$,就称为全行为(all-action)方法,因为在某个时间步估算梯度时,必须考虑所有可能执行的行为. 在实际的算法中,经常采用单个行为方法(single-action),也就是在某个时间步估计梯度时,仅仅考虑该时刻实际执行的那个行为,即

$$\frac{\partial \eta}{\partial \theta} \approx \frac{\partial \pi}{\partial \theta}Q^{\pi}(s_t, a_t). \quad (11)$$

单个行为方法中,为了防止过于频繁地选择某一个行为而忽略其他行为,往往还除以$\pi(a|\theta, s)$:

$$\frac{\partial \eta}{\partial \theta} \approx \frac{\partial \pi}{\partial \theta}Q^{\pi}(s_t, a_t)\frac{1}{\pi(s,a)}. \quad (12)$$

后面的描述中,将不会过分注重区分全行为和单个行为方法,尽管二者在实际的执行过程中有不同之处,但是从理论上来说,二者的区别不大.

#### 2.1.2 策略梯度算法的收敛条件

为了保证策略梯度算法收敛,必须满足以下几个条件:

1) 对任何的状态$s$,行为$a$和参数$\theta_i$,$\frac{\partial \pi(s,a)}{\partial \theta_i}$存在,并且$\frac{\partial \pi(s,a)}{\partial \theta_i}\frac{1}{\pi(s,a)}$一致有界.

2) 对任何的状态$s$,回报$r(s)$一致有界.

3) 针对任何一组参数$\theta \in R^n$所对应的状态转移矩阵$P(\theta)$,具有唯一的平稳分布$\lambda(\theta)$,满足如下的平衡方程:

$$\lambda(\theta)P(\theta) = \lambda(\theta).$$

现在问题的关键就是如何得到$Q^{\pi}(s,a)$. 因为$Q^{\pi}(s,a)$往往是不知道的,需要对它的值进行估计. 其实,所有的策略梯度方法,不同之处仅仅在于估算方法的不同. 下面结合各种基本的策略梯度算法$Q^{\pi}(s,a)$来证明这个论断.

### 2.2 各种基本算法

#### 2.2.1 REINFORCE算法

最直接的想法就是把实际得到的回报$R_t = \sum_{k=1}^{\infty}\gamma^{k-1} r_{t+k}$来作为$t$时刻的的$Q^{\pi}(s_t, a_t)$近似值. 这种算法就是Williams提出的REINFORCE算法[18],这是在增强学习领域最早的策略梯度算法:

$$\frac{\partial \eta}{\partial \theta} \propto \frac{\partial \pi(s_t, a_t)}{\partial \theta} R_t \frac{1}{\pi(s_t, a_t)}. \quad (13)$$

作为最早被提出的策略梯度方法,REINFORCE算法并没有引起太多的关注,主要原因就是该算法只能处理周期性的马氏决策问题,并且收敛速度很慢.

#### 2.2.2 带有值函数逼近器的策略梯度方法

如果状态是连续的情况,那么此时,就需要函数逼近器来对$Q$进行逼近. 但是,函数逼近器必须满

足一定的条件,才能保证算法的收敛[8,19].

假设利用具有参数 $w$ 的函数 $f_w: S \times A \to R$ 来逼近 $Q$,并且参数 $w$ 满足条件:

$$w = \arg\min_w \sum_t (\hat{Q}_t^{\pi} - f(s_t, a_t))^2. \quad (14)$$

式中:$\hat{Q}_t^{\pi}$ 为 $Q_t^{\pi}$ 的某种估计值,比如可以是 $R_t$. 如果函数 $f$ 满足式(14),同时又满足相容性条件:

$$\frac{\partial f_w(s,a)}{\partial w} = \frac{\partial \pi(s,a)}{\partial \theta}\frac{1}{\pi(s,a)}. \quad (15)$$

那么

$$\frac{\partial \eta}{\partial \theta} = \sum_s d^{\pi}(s) \sum_a \frac{\partial \pi}{\partial \theta} f_w(s,a). \quad (16)$$

式(16)的证明过程,请参看文献[8].

关于值函数逼近器的选择,一种可行的方案是采用如下的形式:

$$f_w(s,a) = w^{\mathrm{T}} \frac{\nabla_\theta \pi(s,a)}{\pi(s,a)}. \quad (17)$$

由式(17)方法可以很容易地构造出策略迭代方法[8],并且该方法的收敛性也很容易证明. 但是,由于每次更新 actor 的参数之后,critic 的参数需要重新求解,从这个意义上来说,critic 的参数并不具有"学习"功能,并且求解 critic 参数的过程可能也比较耗时,这就影响了整个算法的收敛速度.

2.2.3　GPOMDP 算法

Baxter 等提出了 GPOMDP 算法,尽管作者给出了一种比较复杂的证明方法,但是,从迭代过程来看,GPOMDP 仍然是利用式(12)来估计梯度,只不过是行为值函数的估计采用了以下公式:

$$\hat{Q}(s_t, a_t) = \sum_{k=1}^{H-t} \gamma^{k-1} r_{t+k}. \quad (18)$$

下面证明这个结论:

GPOMDP 算法中,迭代公式为

$$z_{t+1} = \gamma z_t + \frac{\partial \pi(s_t, a_t)}{\partial \theta}\frac{1}{\pi(s_t, a_t)}. \quad (19)$$

$$\Delta_{t+1} = \Delta_t + r_{t+1} z_{t+1}. \quad (20)$$

式中:$z_t$ 称为适合度轨迹,初始化为 0 向量,$\Delta_0$ 也初始化为 0 向量. 上述公式迭代 $H$ 次后,梯度的估计值为 $\frac{\partial \eta}{\partial \theta} = \frac{1}{H}\Delta_H$. 为了简化证明过程,只分析 $\Delta_H$.

为了书写简便,定义 $\nabla\pi(s,a) = \frac{\partial \pi(s,a)}{\partial \theta}$.

因为 $z_0 = 0$,所以

$$z_1 = \frac{\nabla\pi(s_0,a_0)}{\pi(s_0,a_0)},$$

$$z_2 = \gamma\frac{\nabla\pi(s_0,a_0)}{\pi(s_0,a_0)} + \frac{\nabla\pi(s_1,a_1)}{\pi(s_1,a_2)},$$

$$z_3 = \gamma^2\frac{\nabla\pi(s_1,a_0)}{\pi(s_0,a_0)} + \gamma\frac{\nabla\pi(s_1,a_1)}{\pi(s_1,a_1)} + \frac{\nabla\pi(s_2,a_2)}{\pi(s_2,a_2)},$$

$$\cdots$$

$$z_H = \gamma^{H-1}\frac{\nabla\pi(s_0,a_0)}{\pi(s_0,a_0)} + \gamma^{H-2}\frac{\nabla\pi(s_1,a_1)}{\pi(s_1,a_1)} + \cdots + \frac{\nabla\pi(s_{H-1},a_{H-1})}{\pi(s_{H-1},a_{H-1})},$$

所以,

$$\Delta_H = \sum_t^H r_t z_t = \sum_t^H \left[\frac{\nabla\pi(s_t,a_t)}{\pi(s_t,a_t)} \sum_k^{H-t} \gamma^{k-1} r_{t+k}\right]. \quad (21)$$

此时可以看出,估计式 $\frac{1}{H}\Delta_H$ 就是在 $\hat{Q}^{\pi}(s_t,a_t) = \sum_{k=1}^{H-t}\gamma^{k-1} r_{t+k}$ 前提下,对梯度估计 $H$ 次的平均值.

Baxter 等分析了折扣因子 $\gamma$ 的取值对上述算法的影响:$\gamma$ 越接近于 1,梯度估计越精确,事实上,估计的偏差正比于 $(1-\gamma)$,但是,$\gamma \to 1$ 时,方差也无限增大,所以,梯度估计的精确性与方差是一对矛盾,必须选择适当的 $\gamma$ 对二者进行折衷处理. 因此,尽管可以通过减小 $\gamma$ 来减小方差,但是,这种方法的调整能力有限.

GPOMDP 解除了 REINFORCE 算法只能求解周期性马氏决策问题的限制,但是仍然没能提高算法的收敛速度.

2.3　现有策略梯度算法的推广

根据时域差分的思想,在策略梯度算法中可以利用截断回报作为 $Q(s,a)$ 的近似值,例如,$t$ 时刻的 $Q(s_t,a_t)$ 可以用一步截断回报来估计

$$Q(s,a) \approx R_t^{(1)} = r_{t+1} + \gamma\hat{Q}(s_{t+1},a_{t+1}). \quad (22)$$

式中:$\hat{Q}$ 为 $Q$ 的某种估计值. 也就是利用 $t+1$ 时刻的估计值,来对 $t$ 时刻的估计值进行更新.

把上式推广到 $k$ 步截断回报为

$$Q(s,a) \approx R_t^{(k)} = r_{t+1} + \lambda r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k \hat{Q}(s_{t+k},a_{t+k}). \quad (23)$$

当 $k = \infty$ 时,就是实际回报

$R_t^{(\infty)} = R_t = \sum_{k=1}^{\infty}\gamma^{k-1} r_{t+k}$. 因此,REINFORCE 算法是截断回报算法的一种特例.

继续推广,可以利用各种截断回报的加权和来 $Q(s,a)$ 近似

$$Q(s,a) \approx R_t^{a} = \sum_k^H \alpha_k R_t^{(k)}. \quad (24)$$

只要是权系数满足 $\sum_{k=1}^{H}\alpha_k = 1$,则 $R_t^a$ 为 $Q(s,a)$ 的无偏估计. 对比式(18)和式(24),可以看出,GPOMDP 其实是式(24)在 $\alpha_k = \frac{1}{H}$ 时的一个特例.

当然也可以利用 $\lambda$ 回报来计算 $Q(s,a)$ 近似值:

$$Q(s,a) \approx R_t^{\lambda} = (1-\lambda)\sum_{k}^{\infty}\lambda^{k-1}R_t^{(k)}. \quad (25)$$

## 3 提高收敛速度的几种方法

策略梯度方法最大的问题就是收敛速度太慢,因此,提高策略梯度方法的收敛速度,是该领域目前的研究热点. 尽管近年来涌现了各种各样的算法. 总的来说,主要分为3类,下面逐一进行介绍.

### 3.1 自然梯度方法

常规的梯度方法,估计的是$\frac{\partial\eta}{\partial\theta}$,然后利用该梯度方向更新参数. 但是,这个方向并不一定是指标函数增加最快的方向[20]. 为了说明这个问题,需要定义2个概念.

**定义7** 参数之间的距离 假设参数$\theta$,给它一个很小的扰动$\mathrm{d}\theta$,如果参数空间是具有正交基的欧式空间,那么,$\theta+\mathrm{d}\theta$和$\theta$的距离为

$$\|\mathrm{d}\theta\|^2 = \sum_{i}^{n}(\mathrm{d}\theta)^2. \quad (26)$$

但是,对于具有非正交基的空间,其距离为

$$\|\mathrm{d}\theta\|^2 = \sum_{i,j}g_{i,j}(\theta)\mathrm{d}\theta_i\mathrm{d}\theta_j. \quad (27)$$

实际上,式(27)是式(26)的推广,对于具有正交基的欧式空间:

$$g_{i,j}(\theta) = \begin{cases} 1 & i=j, \\ 0 & i\neq j. \end{cases}$$

式(27)就成了式(26). 定义了参数之间的距离,就可以定义指标函数增加最快的方向.

**定义8** 指标函数$\eta$增加最快的方向 当$\|\mathrm{d}\theta\|^2$固定为一个很小的正数$\varepsilon^2$,使得$|\eta(\theta+\mathrm{d}\theta|$最大的$\mathrm{d}\theta$,就是使得指标函数增加最快的方向.

如果,参数空间是黎曼空间,那么,参数距离应该是式(27),此时,使得指标函数增加最快的方向不再是常规梯度方向,而是自然梯度(natural gradient)方向. 如果定义矩阵$G=(g_{i,j})$,那么,根据文献[20],有下面的定理:

**定理2**:优化指标相对于策略参数的自然梯度为

$$\frac{\partial\eta}{\partial\theta_N} = G^{-1}\frac{\partial\eta}{\partial\theta} \quad (28)$$

Amari等证明了上述定理. 并指出,具有多层感知器的神经网络的参数空间是黎曼空间[20]. 基于上述分析,kakade提出了自然策略梯度方法[21],以及Perters等提出了NAC算法(natural actor-critic)[22].

尽管自然梯度方法比常规的梯度方法收敛速度要快,但是,求解自然梯度时,仍然需要先估计常规的梯度,所以利用自然梯度的方法,并没有解决梯度估计过程中方差过大的问题.

### 3.2 方差减小方法

策略梯度方法一个缺陷就是在梯度估计的过程中方差过大,这影响了算法的收敛速度,成为策略梯度方法实用化的一个很大障碍. 因此,如何减小梯度估计过程中的方差,就成为了一个研究重点. 一般来说,有2类方法可以减小方差:回报基线方法和结合值函数方法. Greensmith详细论述了减小方差的2种方法[23].

#### 3.2.1 回报基线方法

事实上,在式(10)中增加一个常数不会影响梯度估计的期望值,也就是

$$\frac{\partial\eta}{\partial\theta} = \sum_{s}d^{\pi}(s)\sum_{a}\frac{\partial\pi(s,a)}{\partial\theta}Q^{\pi}(s,a) = \sum_{s}d^{\pi}(s)\sum_{a}\frac{\partial\pi(s,a)}{\partial\theta}(Q^{\pi}(s,a)-B). \quad (29)$$

式中:B为常数,称为回报基线.

为了证明上式,只需证明:

$$\sum_{a}\frac{\partial\pi}{\partial\theta}Q^{\pi}(s,a) = \sum_{a}\frac{\partial\pi}{\partial\theta}(Q^{\pi}(s,a)-B).$$

因为$\pi(s,a)$为概率,所以满足$\sum_{a}\pi(s,a)=1$,因此

$$\sum_{a}\frac{\partial\pi}{\partial\theta}(Q^{\pi}(s,a)-B) = \sum_{a}\frac{\partial\pi}{\partial\theta}Q^{\pi}(s,a) - B\sum_{a}\frac{\partial\pi}{\partial\theta} = \sum_{a}\frac{\partial\pi}{\partial\theta}Q^{\pi}(s,a) - B\frac{\partial\sum_{a}\pi(s,a)}{\partial\theta} = \sum_{a}\frac{\partial\pi}{\partial\theta}Q^{\pi}(s,a) - B\frac{\partial 1}{\partial\theta} = \sum_{a}\frac{\partial\pi}{\partial\theta}Q^{\pi}(s,a).$$

回报基线的引入,不会改变期望,但是会影响方差,有可能使得方差增大,也可能使得方差减小,使得方差最小的那个值,称为最优回报基线[15]. 因此,在利用回报基线的算法中,一定要找到这个最优回报基线,才能有效地减小方差. Weaver等把回报基线引入到GPOMDP算法中,并且求出了最优的回报基线[24]. 王学宁等把回报基线应用到求解部分可观测的马氏决策过程中,取得了较好的效果[15]. Munous提出的几何方法,其实也是一种基线方法[25].

#### 3.2.2 结合值函数方法

结合值函数方法,也称执行器-评判器(actor-critic)算法,可以融合策略梯度方法和值函数方法的优点. 其中,actor表示策略,用来选择行为. critic利用一个函数逼近器来学习值函数,用来对策略进行评价. 在这个体系中,actor的参数只要是基于梯度方法来更新的,那么就有可能收敛到局部最优,因此actor-critic算法保留了策略梯度算法的收敛性.

同时,由于值函数可以减小估计过程的方差,因此,又具有较好的收敛速度[19].

假设 critic 的函数逼近器采用线性基函数形式:

$$Q_r^{\theta}(s,a) = \sum_{j}^{m} r^j \phi_{\theta}^j(s,a). \tag{30}$$

式中:$r=(r^1, r^2, \cdots, r^m) \in R^m$ 表示 critic 的参数. 基函数根据 actor,也就是策略的参数构造出来,如果令:

$$\phi_{\theta}^j(s,a) = \frac{\partial \pi(s,a)}{\partial \theta_j} \frac{1}{\pi(s,a)}. \tag{31}$$

构造基函数的一种最简单的方法就是令 $\phi_{\theta}^j(s,a) = \phi_{\theta}^j(s,a)$. 可以看出,actor-critic 算法和前面提到的策略迭代算法在结构上完全相同,但是 actor-critic 算法与前面提到的策略迭代方法最根本的区别是:actor-critic 算法在每一个时间步,都利用时域差分(temporal-difference)来更新 critic 的参数 $r$,利用式(12)来更新 actor 的参数. 而策略迭代算法是求解值函数逼近器的参数时,策略参数不动,以保证策略的平稳性. 当根据值函数调整了策略参数之后,值函数的参数需要重新求解. 因此,actor-critic 算法较策略迭代算法有了改进,在收敛速度上也有了提高,已经被应用到实际的系统中[26].

### 3.3　局部搜索的策略梯度估计方法

REINFORCE 算法的一个缺陷就是梯度估计的过程中方差过大. Grudic 等指出,方差过大的原因就是每当访问一个状态时,策略梯度算法都要进行梯度计算. 策略梯度算法的核心是根据在每一个状态下选择不同行为对优化指标的影响不同来调整参数. 因此,REINFORCE 算法收敛的一个必要条件就是需要对所有状态进行无限遍历,并且在访问同一个状态时,最好是采用不同的行为,这就导致了计算过程中的方差过大[27]. 基于上述分析,Grudic 等在式(32)、(33)的基础之上,构造出了 ATPG 算法:

$$\frac{\partial \eta}{\partial \theta} = \sum_{s} d^{\pi}(s) \sum_{a} \frac{\partial \pi(s,a)}{\partial \theta} (Q^{\pi}(s,a) - V^{\pi}(s)). \tag{32}$$

$$V^{\pi}(s) = \frac{1}{M} \sum_{j}^{M} Q^{\pi}(s,a^j). \tag{33}$$

式中:$M$ 表示在状态 $s$ 时可以选择的行为的个数. 式(32)比基本的梯度公式(10)多了一项 $V^{\pi}(s)$,这一项可以看成是回报基线,因此,并不影响梯度的估计. ATPG 算法如下:令

$$P_t = \frac{\partial \pi(s_t,a_t)}{\partial \theta} q_t + \frac{\partial \pi(s_{t+1},a_{t+1})}{\partial \theta} q_{t+1}. \tag{34}$$

式中:

$$q_t = \frac{1}{M} \frac{Q^{\pi}(s_t,a_t) - \overline{Q}_t}{\pi(s_t,a_t)\pi(s_{t+1},a_{t+1})}. \tag{35}$$

$$q_{t+1} = \frac{1}{M} \frac{Q^{\pi}(s_{t+1},a_{t+1}) - \overline{Q}_t}{\pi(s_t,a_t)\pi(s_{t+1},a_{t+1})}. \tag{36}$$

而

$$\overline{Q}_t = \frac{1}{2}\left( Q^{\pi}(s_t,a_t) + Q^{\pi}(s_{t+1},a_{t+1}) \right). \tag{37}$$

如果令 $Q^{\pi}(s_t,a_t) = \sum_{k=1}^{H-t} \gamma^{k-1} r_{t+k}$,式中:$H$ 表示一个周期内的时间步数.

则 $$\frac{\partial \eta}{\partial \theta} = \sum_{t=1}^{H} \phi_t.$$

而 $\phi_t = \begin{cases} P_t, & a_t \neq a_{t+1}, \\ 0, & \text{otherwise}. \end{cases}$

算法的基本出发点为:只是在行为改变的时候才估计梯度,并且只关心 $Q^{\pi}(s_t,a_t)$ 和 $Q^{\pi}(s_{t+1},a_{t+1})$ 相对差异. Grudic 证明了上述算法的收敛性[27].

### 3.4　其他改进算法

Schraudolph 等提出的利用增益向量自适应的方法(SMDPOMDP)来提高策略梯度算法的收敛速度. 仿真实验表明,SMDPOMDP 方法比 GPOMDP 和自然梯度策略梯度方法收敛的都要快[28].

Bagnell 等提出了一种基于核的非参数化策略搜索算法. 这个算法可以看作是 REINFORCE 算法的推广[29]. 基于核的策略搜索算法,可以很好地把增强学习和核方法结合起来. 随着该类算法的进一步发展,有望得出高效的算法.

## 4　其他策略搜索方法

### 4.1　PEGASUS 算法

PEGASUS 算法不同于上述的各种算法[30-31],它没有利用式(10)进行梯度估计,而是通过预先设定一组随机变量,把随机问题转化为确定性的问题来对问题求解. 文献[31]证明了下面的定理:

**定理 3**:给定某个满足定义 1 的 MDP 模型为 $M$,以及策略空间 $\prod$,可以构造出一个确定性的模型 $M'$,及其策略空间 $\prod'$.

该确定性的模型的状态转移为确定的,而非随机的. 给定状态—行为对 $(s,a)$ 的情况下,下一时刻的状态是根据状态概率来确定的. 在计算机的仿真过程中,往往是首先产生一个在之间均匀分布的随机数 $p$,然后根据这个随机数 $p$ 以及状态转移概率来决定下一时刻的状态. 也就是说有了这个随机数 $p$,下一时刻的状态就是完全确定的了. 为了构造确定性的模型 $M'$,这里定义一个确定性的转移函数:

$$g: S \times A \times p \to S$$

根据 MDP 模型为 $M$,构造确定性模型 $M$ 的过程如下:

行为空间不变,状态由原来的 $S$ 变成 $S': S \times [0,1]^{\infty}$,也就是说,在 $M$ 中,一个状态为向量$(s, p_1, p_2, \cdots)$,$s$ 为 $M$ 的状态. 例如,假设在 $t$ 时刻状态为 $(s_t, p_1, p_2, \cdots)$,行为为 $a_t$,则 $t+1$ 时刻的状态为 $(s_{t+1}, p_2, p_3, \cdots)$,式中:$s_{t+1} = g(s_t, a_t, p_1)$. 对于某个策略 $\pi \in \prod$,其在策略空间 $\prod$ 的对应策略为 $\pi'(s, p_1, p_2, \cdots) = \pi(s)$. 最后,令 $R'(s, p_1, p_2, \cdots) = R(s)$.

根据上述的构造方法,对于 2 个对应的策略 $\pi \in \prod$ 和 $\pi' \in \prod'$,有下式成立:

$$V_M(\pi) = V_{M'}(\pi'). \tag{38}$$

这也就隐含了关系式 $opt(M, \prod) = opt(M', \prod')$,也就是说,根据模型 $M'$,在策略空间中寻找到的最优策略,其对应策略也是在原问题中的最优策略. 在确定性模型中,因为状态转移是确定性的,所以如果给定策略 $\pi \in \prod'$,状态 $s \in S'$,那么此时刻以后的所有状态也就完全确定了. 因此计算该状态值 $V^{\pi}_{M'}(s)$ 也就相应的简单了. 假设问题有 $m$ 个初始状态,第 $i$ 个初始状态记为 $s_0^{(i)}$,则策略 $\pi$ 的值可用下式计算:

$$V_{M'}(\pi) \approx \frac{1}{m} \sum_{i=1}^{m} V^{\pi}_{M'}(s_0^{(i)}). \tag{39}$$

使得优化指标 $\eta$ 最大的策略,也就是使得 $V(\pi)$ 最大的策略,因此,现在问题转化为寻找最优策略

$$\pi^* = \arg\max_{\pi} V(\pi). \tag{40}$$

为此需要知道 $\frac{\partial V}{\partial \theta}$,求解该梯度有 2 种方法,一种是数值方法,一种是解析方法[31]. 数值方法就是给每个参数一个小小的扰动,根据这个扰动来计算梯度,梯度的第 $i$ 个分量为

$$\frac{\partial V(\theta)}{\partial \theta_i} = \frac{V(\theta + \alpha e_i) - V(\theta - \alpha e_i)}{2\alpha}.$$

式中:$e_i = (0, 0, \cdots, 1, 0, \cdots, 0)$,即第 $i$ 个分量为 1,其他分量为 0 的向量,其维数和 $\theta$ 相同. 解析方法比较繁琐,这里不在赘述,详情请参看文献[31].

相比较而言,PEGAUSU 算法收敛速度比其他算法都快,但是,该算法只能应用到仿真试验中. 目前,已经有利用 PEGASUS 算法成功控制直升机的例子[32]. Strens 等利用成对比较法(paired comparisions)提高了 PEGASUS 算法的性能,并且解决了"过拟合"的问题[33].

### 4.2 本质动力学算法

本质动力学算法实质上是线性二次调节器的推广. 它首先学习系统模型,然后利用该模型计算当前策略的值,并且可以计算策略值相对于策略参数的梯度. 在自行车问题中,EDA 算法比 PEGASUS 收敛要快,并且利用的策略也比较简单[34]. 但是,EDA 算法需要学习系统模型,并且认为系统模型的模型是线性的,或者至少是局部线性的,这就限制了该算法的应用范围.

### 4.3 和 SVM 结合的策略搜索方法

策略梯度方法由于方差过大,导致收敛速度很慢,即使很简单的问题,也要学习很长的时间. 并且,往往初始策略是随机给定的,因此学习的过程中,尤其是学习的早期,学习体会选择一些不正确的行为,在实际的试验中,这些不正确的行为可能会引起各种事故. 基于上述 2 点,王学宁等把 SVM 引入到策略梯度学习方法中来,提出了一种混合式策略梯度增强学习方法[35]. 在该方法中,初始策略不是随机给定的,而是根据先验知识,或者人为控制,产生一些样本点,然后根据这些样本点,利用 SVM 方法产生一个初始策略. 这个初始策略往往是可行的,但不是最优的,因此,再利用策略梯度方法对这个初始策略进行优化. 再利用优化之后的策略产生一些样本点,根据这些样本点,利用 SVM 方法产生一个初始策略,再利用策略梯度方法进行优化,如此循环往复,使得策略达到一个局部最优解.

### 4.4 层次化策略梯度方法

对于大规模的问题,比如说状态和行为都是连续的,直接求解比较困难,甚至是无法求解. 但是如果把问题分解成若干个小问题,往往就能迎刃而解. 这就是层次化增强学习方法的基本思想. 在 Chavamzadeh 等提出层次化策略梯度(Hierarchical Policy Gradient)算法之前[36],已经有很多关于层次化值函数方法的文献[37-39].

Chavamzadeh 等提出的 HPG 算法中,把问题分为 2 层,下层是分解后的小问题,上层是在某一时刻选择需要解决哪个小问题. 利用策略梯度算法进行下层问题的求解,而利用值函数方法进行上层的学习. 该方法可以有效地解决大规模问题.

## 5 结 论

尽管策略梯度增强学习方法从 1992 年已经被提出,但是,到了 2000 年前后,才逐渐引起人们的注意. 虽然刚刚经过几年的发展时间,已经有众多的策略梯度算法被提出. 但是,由于策略梯度方法的收敛速度太慢,导致策略梯度还不能应用到大规模的问题之中. 因此,策略梯度算法中需要研究的就是如何

提高收敛速度.根据前面的分析,提高收敛速度,有以下几类方法:

1)减小方差,相对来说,这类方法研究的已经很多,但是效果有限.

2)利用自然梯度方法,这类方法依然要用到常规的梯度,这就离不开梯度估计算法,仍然不能避免梯度估计过程中的方差过大的问题.

3)利用先验知识.利用先验知识,是策略梯度算法走向实用化的一个很有效并且很重要的手段.但是,如何方便有效地利用先验知识?先验知识的引入是否影响算法的收敛性?先验知识并不能百分之百的正确,如何消除错误的先验知识带来的错误导向?这 3 个问题有待进一步的研究.

4)利用层次化策略梯度方法.对于能分解的问题,这是一个有效的方法.但是并不是所有的问题都可以分解.

利用局部策略梯度估计方法.由于仅仅考虑行为改变时的情况,因此极大地减小了运算量,这类方法虽然目前还没有引起足够的重视,但是有望能较大幅度地提高算法的性能.

## 参考文献:


[1]徐　昕.增强学习及其在移动机器人导航与控制中的应用研究[D].长沙:国防科技大学,2002.
XU Xin. Reinforcement learning and its applications in navigation and control of mobile robots[D]. Changsha: National University of Defence Technology, 2002.

[2]SUTTON R,BARTO A. Reinforcement learning, an introduction[M]. MIT Press, 1998.

[3] SINGH S P. Learning to solve Markovian decision processes[D]. University of Massachusetts, 1994.

[4]ROY B V. Learning and value function approximation in complex decision processes[M]. MIT Press, 1998.

[5]WATKINS C. Learning from delayed rewards[D]. Cambrideg: University of Cambridge, 1989.

[6] HUMPHRYS M. Action selection methods using reinforcement learning[D]. Cambrideg: University of Cambridge,1996.

[7]BERTSEKAS D P, TSITSIKLIS J N. Neuro-dynamic programming[M]. Athena Scientific,Belmont, Mass., 1996.

[8]SUTTON R S, MCALLESTER D, SINGH S, et al. Policy gradient methods for reinforcement learning with function approximation[A]. In: Advances in Neural Information Processing Systems[C]. Denver, USA,2000.

[9]BAIRD L C. Residual algorithms: reinforcement learning with function approximation[A]. In: Proc. Of the 12 # Int. Conf. on Machine Learning[C]. San Francisco, 1995.

[10] TSITSIKLIS J N, ROY V B. Feature-based methods for large scale dynamic programming[J]. Machine Learning,1996(22):59 - 94.

[11]徐　昕,贺汉根.神经网络增强学习的梯度算法研究[J].计算机学报,2003,26(2):227 - 233.
XU Xin, HE Hangen. A gradient algorithm for reinforcement learning based on neural networks[J]. Chinese Journal of Computers, 2003, 26(2): 227 - 233.

[12]BAXTER J, BARTLETT P L. Infinite-horizon policy-gradient estimation[J]. Journal of Artificial Intelligence Research,2001(15):319 - 350.

[13]ABERDEEN D A. Policy - gradient algorithms for partially observable Markov decision processes[D]. Australian National University, 2003.

[14] GREENSMITH P L, BAXTER J. Variance reduction techniques for gradient estimation in reinforcement learning[J]. Journal of Machine Learning Reseach, 2002(4): 1471 - 1530.

[15]王学宁,徐　昕,吴　涛,贺汉根.策略梯度强化学习中的最优回报基线[J].计算机学报,2005,28(6):1021 - 1026.
WANG Xuening, XU Xin, WU Tao, HE Hangen. The optimal reward baseline for policy-gradient reinforcement learning[J]. Chinese Journal of Computers, 2005, 28(6):1021 - 1026.

[16]SCHWARTZ A. A reinforcement learning method for maximizing undiscounted rewards[A]. In Proceedings of the Tenth International Conference on Machine Learning[C]. Morgan Kaufmann,San Mateo,CA,1993.

[17]MAHADEVAN S. To discount or not to discount in reinforcement learning: a case study comparing R-learning and Q-learning[A]. In: Proc. Of International Machine Learning Conf[C]. New Brunswick, USA,1994.

[18] WILLIAMS R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine Learning,1992(8):229 - 256.

[19] KONDA V R, TSITSIKLIS J N. Actor-critic algorithms[J]. Adv. Neural Inform Processing Syst, 2000 (12): 1008 - 1014.

[20]AMARI S. Natural gradient works efficiently in learning [J]. Neural Computation, 1998, 10(2):251 - 276.

[21]KAKADE S. A natural policy gradient[A]. Advances in Neural Information Processing Systems 14[C]. MIT Press, 2002.

[22]PETERS J, VIJAYAKUMAR S,SCHAAL S. Natural actor-critic[A]. In 16th European Conference on Machine Learning (ECML 2005)[C]. [s.l.],2005.

[23] GREENSMITH E. Variance reduction techniques for gradient estimation in reinforcement learning[J]. Jour-

nal of Machine Learning Reseach, 2002(4): 1471 - 1530.

[24]WEAVER L, TAO N. The optimal reward baseline for gradient-based reinforcement learning[A]. Proceedings of 17 # Conference in Uncertainty in Artificial Intelligence[C]. Washington ,2001.

[25]MUNOS R. Geometric variance reduction in Markov chains: application to value function and gradient estimation[J]. Journal of Machine Learning Research, 2006 (7) :413 - 427.

[26]BERENJI H R, VENGEROV D. A convergent actor-critic-based FRL algorithm with application to power management of wireless tansmitters[J]. IEEE Transactions on Fuzzy Systems, 2003, 11(4): 478 - 485.

[27]GRUDIC G Z, UNGER L R. Localizing policy gradient estimates to action transitions[A]. Seventeenth int. Conference on Machine Learning, Stanford University [C]. 2000:343 - 350.

[28]NICOL N S, YU Jin,ABERDEEN D. Fast online policy gradient learning with SMD gain vector adaptation [A]. In Advances in Neural Information Processing Systems (NIPS) [C]. The MIT Press, Cambridge, MA, 2006.

[29]BAGNELL J A, SCHNEIDER J. Policy search in kernel hilbert space[EB/OL]. http://citeseer.ist.psu.edu/650945.html. 2005 - 05 - 27.

[30]NG A, JORDAN M. Pegasus: a policy search method for large MDPs and POMDPs approximation[A]. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence [C]. San Francisco, 2000.

[31]Diplomarbeit. The Reinforcement Learning Toolbox, Reinforcement Learning for Optimal Control Tasks[D], University of Technology, Graz, 2005.

[32]NG A Y, KIM H J, MICHAEL I, SASTRY S. Autonomous helicopter flight via Reinforcement Learning [A]. In Neural Information Processing Systems 16[C]. [s.l.], 2004.

[33]STRENS M J, MOORE A. Policy search using paired comparisons[J]. Journal of Machine Learning Research, 2002(3) :921 - 950.

[34]MARTIN C. The essential dynamics algorithm: fast policy Search in continuous worlds[R]. MIT Media Laboratory, Vsion and Modelling Technical Report. 2004.

[35]WANG X N, XU X, WU T, HE H G, ZHANG M. A hybrid reinforcement learning combined with SVM and its applications[A]. Proceedings of the International Conference on Sensing, Computing and Automation [C]. Chongqing, China, 2006.

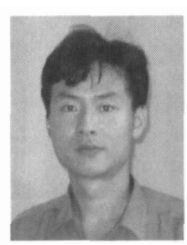
[36]GHAVAMZADEH M, MAHADEVAN S. Hierarchical policy gradient algorithms[A]. In Proceedings of the Twentieth International Conference on Machine Learning[C]. Washington, D.C., 2003.

[37]DIETTERICH T. The MAXQ method for hierarchical reinforcement learning[A]. In: Proceedings of the fifteenth international conference on machine learning[C]. [s.l.],1998.

[38]PARR R. Hierarchical control and learning for Markov decision processes[D]. University of California, Berkley,1998.

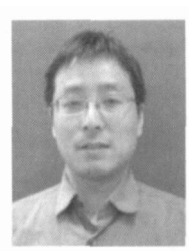
[39]SUTTON R, PRECUP D, SINGH. Between MDPs and Semi-MDPs: a framework for temporal abstraction in reinforcement learning[J]. Artificial Intellignce, 1999(112) :181 - 211.

**作者简介**:

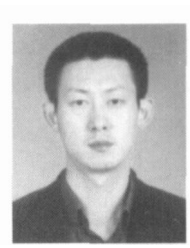


**王学宁**,男,1976 年生,博士研究生,主要研究方向为增强学习、智能控制等,参加国家自然科学基金重点项目一项、青年基金项目一项,863 项目一项,已发表论文 10 余篇,其中被 SCI 收录 3 篇,EI 收录 5 篇.

E-mail:wxn9576@yahoo.com.cn



**陈 伟**,男,1976 年生,博士研究生,主要研究方向为机器人定位与见图、机器学习等,参加国家自然科学基金重点项目一项.



**张 锰**,男,1972 年生,2001 年毕业于国防科技大学计算机学院,获硕士学位. 主要研究方向为指挥自动化. 曾获全军科技进步二等奖 2 项,全军科技进步三等奖 3 项,并在国内外科技期刊上发表论文 12 篇,其中 SCI 检索 1 篇,EI 检索 3 篇.