Pei (https://blogs.cuit.columbia.edu/zp2130/)

Search

Search

+ Add post (https://blogs.cuit.columbia.edu/zp2130/wp-admin/post-new.php)

Menu

- Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/)
- Posts (https://blogs.cuit.columbia.edu/zp2130/posts/)
- Resources (https://blogs.cuit.columbia.edu/zp2130/resources/)
- Cacti-based Framework (https://blogs.cuit.columbia.edu/zp2130/cacti/)
- Publications (https://blogs.cuit.columbia.edu/zp2130/publications/)

Email Address:

zp2130@caa.columbia.edu (mailto:zp2130@caa.columbia.edu)

p@caa.columbia.edu (mailto:p@caa.columbia.edu)

Blog Stats

137,045 hits

State Action/Control

blogs.cuit.columbia.edu/p (https://blogs.cuit.columbia.edu/p/)

Meta

Site Admin (https://blogs.cuit.columbia.edu/zp2130/wp-admin/)

Log out (https://blogs.cuit.columbia.edu/zp2130/wp-login.php?action=logout&_wpnonce=e00c291433)

Entries feed (https://blogs.cuit.columbia.edu/zp2130/feed/)

Comments feed (https://blogs.cuit.columbia.edu/zp2130/comments/feed/)

WordPress.org (https://wordpress.org/)

Actor-Critic Algorithms

Actor-Critic Algorithms (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/Actor-Critic-Algorithms.pdf)

Math Analysis

Methods that learn approximations to both **policy** and **value** functions are often called **actor–critic** methods, where '**actor**' is a reference to the **learned policy**, and '**critic**' refers to the **learned value function**, usually a **state-value function**.

这篇论文提出和分析了一类**actor–critic**算法，用于一参数化系列的随机平稳策略的马尔可夫决策过程（MDP）的基于仿真的优化。

Critic：一个线性近似架构的**TD学习, value function。**

Actor：使用Critic提供的信息，在一个**近似梯度**方向更新。决策, **policy。**

大多数强化学习和神经动态编程方法主要属于以下两类中的一类：

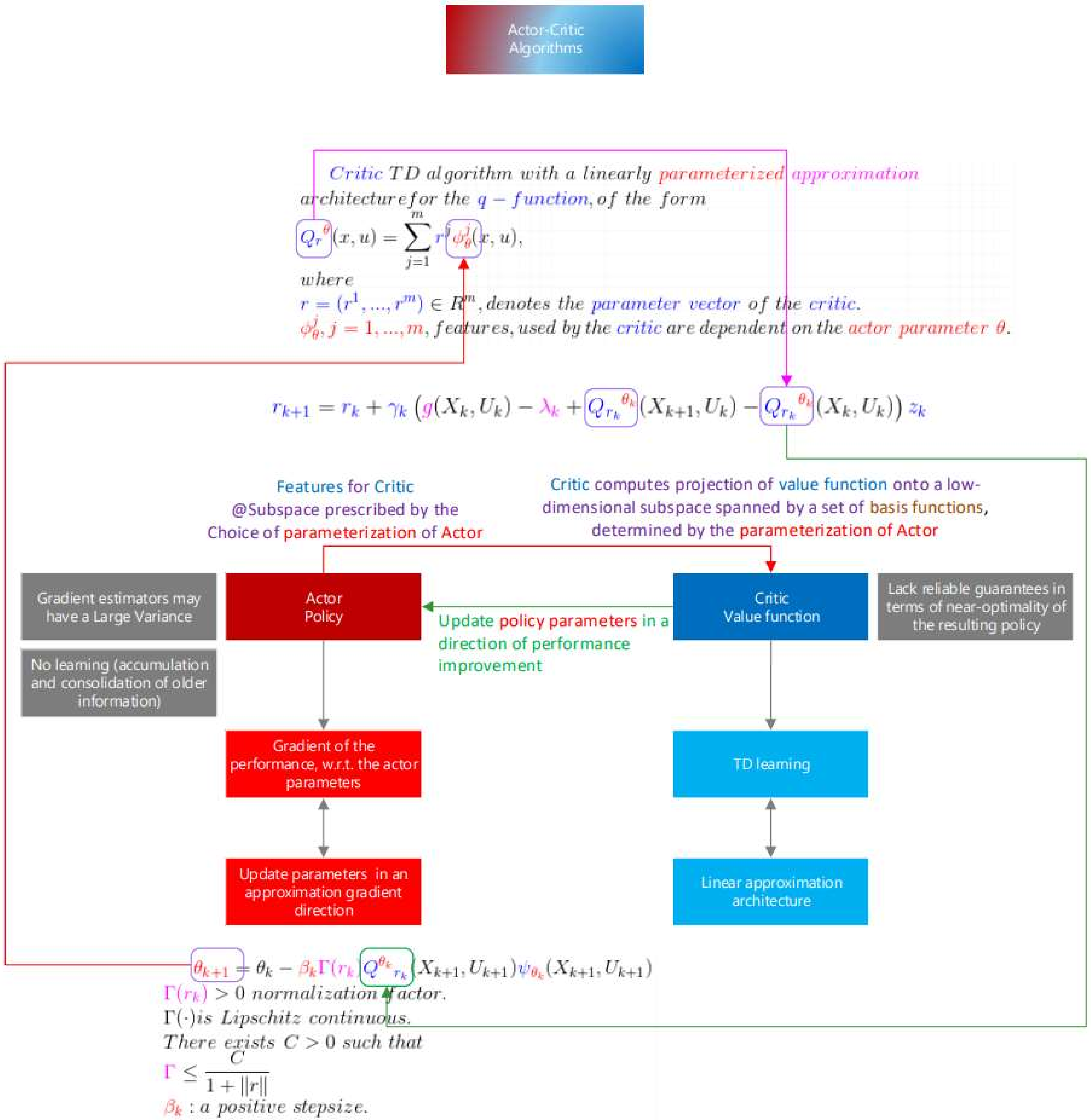
（a）**Actor-only**，**一系列参数化的策略。**性能梯度，对于actor参数的偏导，在提高方向上更新参数。可能的缺点是**梯度估计者可能有大偏差**，而且，策略改变后，新策略估计与过去估计无关，所以，没有"学习"，也就是**没有积累和固化老信息。**

（b）**Critic-only**，**只依赖近似值函数**，目的是学习Bellman公式的近似解，即希望规定一个近-优化的策略。这个方法**不是在策略空间直接优化。**这个方法可能可以找到值函数的"好的"近似函数，**但是在结果策略的近-优化方面缺乏可信度。**

Actor–critic方法结合了actor-only和critic-only的优点。**Critic使用一个近似架构和仿真学习值函数**，然后用来在性能提高方向上**更新actor策略参数**，这个方法是基于梯度，可以得到希望的收敛特性，critic-only只有在非常有限的设置才能保证收敛。对比**actor-only**方法，这种方法能更快地收敛，因为**偏差降低**了。

这篇论文提出actor-critic 算法，证明收敛。这算法基于重要的观察。因为actor的需要更新的参数数量相对状态数量来说很小，critic不用计算或近似准确的高维对象的值函数。实际上，**critic**理想化地计算**值函数**"投影"到一个低维的子空间，即一组完全由**actor参数化**决定的"基本函数"所在的子空间。

Download Actor-Critic Algorithms Flowchart (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/Actor-Critic_Algorithms-2.pdf)



Notation

S : MDP finite **state** space

A : MDP finite **action** space

g : S x A → R given **cost function**

μ : randomized stationary **policy** (RSP), mapping μ that assigns to each **state x** a probability distribution over the **action space A**

$P = \{\mu_\theta; \theta \in R^n\}$: a set of randomized **stationary policies**

θ : parameter in stationary **policy**

$\mu_\theta(x, u)$: probability of **taking action u** when the **state x** is encountered under the **policy** corresponding to **θ**

$\theta \mapsto \mu_\theta(x, u)$: map **probability**, **policy**. 在有关 ***θ*** 的策略下，**状态x**，发生**动作u**的概率。

$p_{xy}(u)$: probability that the next state is y, given that the **current state is x** and the **current action is u**.

$\{X_n\}$: sequence of **states**.

$\{X_n, U_n\}$: **state–action** pairs, MDP, **S** x **A**.

$\psi_\theta(x, u)$: R^n **valued function**

$\nabla\mu_\theta(x, u) = \mu_\theta(x, u)\psi_\theta(x, u)$: **policy gradient**

$\theta \mapsto \psi_\theta(x, u)$: map **value function**

$\pi_\theta(x)$: Markov chains $\{X_n\}$ and $\{X_n, U_n\}$ **stationary probability**. **状态平稳分布的概率。**

$\eta_\theta(x, u) = \pi_\theta(x)\mu_\theta(x, u)$: **stationary probability. 在状态平稳分布的情况下，某状态**发生**动作u的概率。**

$\lambda(\theta) = \sum_{x \in S, u \in A} g(x, u)\eta_\theta(x, u)$: average **cost function**. $R^n \mapsto R$

$\pi_\theta(x)$: Markov chains $\{X_n\}$ **stationary probability**. **状态平稳分布的概率。**

$\mu_\theta(x, u)$: probability of **taking action u** when the **state x** is encountered under the **policy** corresponding to **θ**

$\eta_\theta(x, u) = \pi_\theta(x)\mu_\theta(x, u)$: Markov chains $\{X_n, U_n\}$ **stationary probability.** 在**状态平稳分布**的情况下，**某状态发生动作u的概率。**

$\psi_\theta(x, u)$ $$\begin{aligned}\psi_\theta(x, u) &= \frac{\mu_\theta(x, u)\psi_\theta(x, u)}{\mu_\theta(x, u)} \\ &= \frac{\nabla\mu_\theta(x, u)}{\mu_\theta(x, u)} \\ &= \nabla ln\mu_\theta(x, u)\end{aligned}$$: R^n **valued function**

$\nabla\mu_\theta(x, u) = \mu_\theta(x, u)\psi_\theta(x, u)$: **policy gradient，这个有关策略梯度的定义很巧妙，它将概率与值函数联系在一起。**

$\lambda(\theta) = \sum_{x \in S, u \in A} g(x,u)\eta_\theta(x,u) = \sum_{x \in S, u \in A} g(x,u)\pi_\theta(x)\mu_\theta(x,u)$: average **cost function**. $R^n \mapsto R$

$q_\theta(x,u) = g(x,u) - \lambda(\theta) + \sum_y V_\theta(y)p_{xy}(u)$: **q-function**

$$\begin{aligned}\psi_\theta(x,u) &= \frac{\mu_\theta(x,u)\psi_\theta(x,u)}{\mu_\theta(x,u)} \\ &= \frac{\bigtriangledown \mu_\theta(x,u)}{\mu_\theta(x,u)} \\ &= \bigtriangledown ln \mu_\theta(x,u)\end{aligned}$$

$$\begin{aligned}\lambda(\theta) + V_\theta(x) &= \sum_{x \in S, u \in A} g(x,u)\eta_\theta(x,u) + V_\theta(x) \\ &= \sum_{x \in S, u \in A} g(x,u)\pi_\theta(x)\mu_\theta(x,u) + \sum_{x \in S, u \in A} \sum_y V_\theta(y)\pi_\theta(x)\mu_\theta(x,u)p_{x,y}(u) \\ &= \sum_{x \in S, u \in A} \mu_\theta(x,u)\pi_\theta(x)\left[g(x,u) + \sum_y V_\theta(y)p_{x,y}(u)\right] \\ &= \sum_{u \in A} \mu_\theta(x,u)\left[g(x,u) + \sum_y V_\theta(y)p_{x,y}(u)\right]\end{aligned}$$

$V_\theta(x)$: can be viewed as the “disadvantage” of **state x,** it is the expected **excess cost** – on top of the average cost – incurred if we start at state x. 作用与MDP值函数，总或打折cost 相似。

Theorem 1

$$\frac{\partial}{\partial \theta_i}\lambda(\theta) = \sum_{x,u} \eta_\theta(x,u) q_\theta(x,u) \psi_\theta^i(x,u)$$

where

$\psi_\theta^i(x,u) : the\ ith\ component\ of\ \psi_\theta$

内积： $Define\ the\ inner\ product\ \langle \cdot, \cdot \rangle_\theta\ of\ 2\ real\ valued\ functions\ q_1, q_2\ \langle q_1, q_2 \rangle_\theta = \sum_{x,u} \eta_\theta(x,u) q_1(x,u) q_2(x,u)$

so

$$\frac{\partial}{\partial \theta_i}\lambda(\theta) = \langle q_\theta, \psi_\theta^i \rangle_\theta,\ \ i = 1, ..., n.$$

范数： $||\cdot||_\theta$

将学习q_θ 转化为 q_θ 在子空间 Ψ ***投影***的学习。

$let \prod_\theta : R^{|S||A|} \mapsto \Psi_\theta be\ the\ projection\ operator \prod_\theta q = arg\ min_{\hat{q}\in\Psi_\theta} ||q-\hat{q}||_\theta\ since\ \langle q_\theta, \psi_\theta\rangle_\theta = \left\langle \prod_\theta q_\theta, \psi_\theta \right\rangle_\theta\ it\ is\ enough\ to\ compute(learn)\ the\ projection\ of\ q_\theta\ onto\ \Psi_\theta.$

结论：计算“学习”**值函数**在子空间的**投影**足够了。

Actor-critic algorithms

我们把Actor-critic算法看成在**actor参数空间**的**随机梯度算法**，当actor参数向量是θ，**critic**的工作就是计算 $\prod_\theta q_\theta$ 在子空间Ψ_θ的**投影的近似值**，**actor**用这个**投影的近似值**在**近似梯度方向**更新它的**策略。**

在这篇论文的算法中，需要改变**控制策略control policy**与**特征向量feature vectors**，因为**actor**更新它的**参数。**

Critic

论文里面描述了两个actor-critic 算法，区别只在于critic更新的不同。critic是一个TD算法，q-函数的线性参数近似架构：

$Critic\ TD\ algorithm\ with\ a\ linearly\ parameterized\ approximation architecture for\ the\ q-function, of\ the\ form Q_r^\theta(x,u) = \sum_{j=1}^m r^j\phi_\theta^j(x,u), where r = (r^1,...,r^m) \in R^m, denotes\ the\ parameter\ vector\ of\ the\ critic. \phi_\theta^j, j=1,...,m, features, used\ by\ the\ critic\ are\ dependent\ on\ the\ actor\ parameter\ \theta. \Phi_\theta\ contains\ \Psi_\theta.$

$$\lambda_{k+1} = \lambda_k + \gamma_k(g(X_k,U_k) - \lambda_k)$$
$$r_{k+1} = r_k + \gamma_k\left(g(X_k,U_k) - \lambda_k + Q_{r_k}{}^{\theta_k}(X_{k+1},U_k) - Q_{r_k}{}^{\theta_k}(X_k,U_k)\right) z_k$$

$\gamma_k:\ a\ positive\ stpesize\ parameter$

两个**critic**方法区别只在于**更新z_k 方式不同。**

TD(1) **Critic**: Let x* be a state in S.

$$\begin{aligned} z_{k+1} &= z_k + \phi_{\theta_k}(X_{k+1},U_{k+1}), if\ X_{k+1} \neq x*, \\ &= \phi_{\theta_k}(X_{k+1},U_{k+1}),\ otherwise. \end{aligned}$$

TD(α) **Critic**, $0 \leq \alpha < 1$:

$z_{k+1} = \alpha z_k + \phi_{\theta_k}(X_{k+1},U_{k+1})$

Actor

Finally, the actor updates its parameter vector by letting

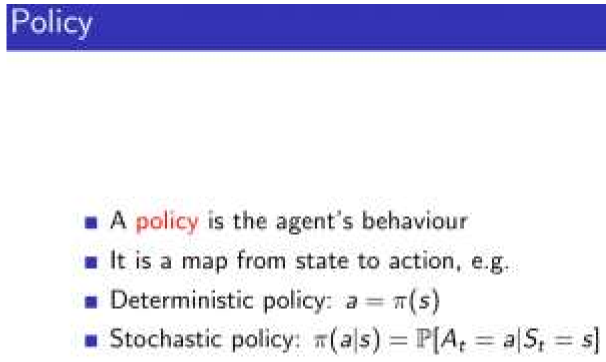
$$\theta_{k+1} = \theta_k - \beta_k\Gamma(r_k)Q^{\theta_k}{}_{r_k}(X_{k+1},U_{k+1})\psi_{\theta_k}(X_{k+1},U_{k+1})\Gamma(r_k) > 0\, normalization\ factor. \Gamma(\cdot) is\ Lipschitz\ continuous. There\ exists\ C>0\ such\ that \Gamma \leq \frac{C}{1+||r||}\beta_k : a\ positive\ stepsize.$$

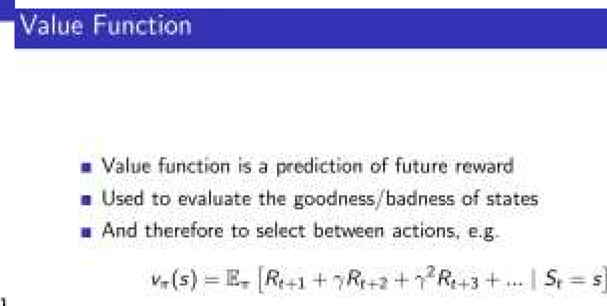
Convergence of actor-critic algorithms

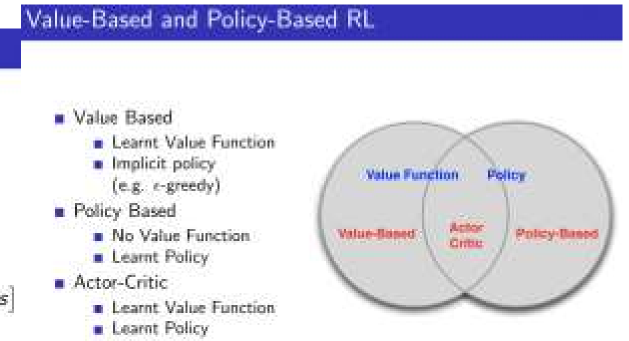
actor–critic算法是基于梯度，不能证明全局优化策略是收敛的，证明cost $\bigtriangledown\lambda(\theta)n \to 0$。

因为$\frac{\beta_k}{\gamma_k} \rightarrow 0$, 对比**critic**更新的尺寸，**actor**的尺寸更新可以忽略不计，所以，当考虑**critic**的时候，**actor是平稳的。**也就是说**解决了actor-only偏差大的问题。**

Policy Gradient vs Q-learning (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_q-learning/)







---

edit (https://blogs.cuit.columbia.edu/zp2130/wp-admin/post.php?post=4493&action=edit)

Author: Z Pei (https://blogs.cuit.columbia.edu/zp2130/author/zp2130/) on February 17, 2019

Categories: Actor-Critic Algorithms (https://blogs.cuit.columbia.edu/zp2130/category/actor-critic-algorithms/), AI (https://blogs.cuit.columbia.edu/zp2130/category/ai/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/category/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/category/rl/)

Tags: Actor-Critic Algorithms (https://blogs.cuit.columbia.edu/zp2130/tag/actor-critic-algorithms/), AI (https://blogs.cuit.columbia.edu/zp2130/tag/ai/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/tag/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/tag/rl/)

Other posts

Policy Gradient Methods for Reinforcement Learning with Function Approximation (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_methods_for_reinforcement_learning_with_function_approximation/) «» Policy Gradient (https://blogs.cuit.columbia.edu/zp2130/policy_gradient/)

Last posts

- Symbolic Netlist to Innovus-friendly Netlist (https://blogs.cuit.columbia.edu/zp2130/symbolic_netlist_to_innovus-friendly_netlist/)
- Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms (https://blogs.cuit.columbia.edu/zp2130/finite-sample_convergence_rates_for_q-learning_and_indirect_algorithms/)
- Solving H-horizon, Stationary Markov Decision Problems In Time Proportional To Log(H) (https://blogs.cuit.columbia.edu/zp2130/paul_tseng_1990/)
- Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear (Sometimes Sublinear) Run Time (https://blogs.cuit.columbia.edu/zp2130/randomized_linear_programming_solves_the_discounted_markov_decision_problem_in_nearly-linear_sometimes_sublinear_run_time/)
- KL Divergence (https://blogs.cuit.columbia.edu/zp2130/kl_divergence/)
- The Asymptotic Convergence-Rate of Q-learning (https://blogs.cuit.columbia.edu/zp2130/the_asymptotic_convergence-rate_of_q-learning/)

- Hierarchical Apprenticeship Learning, with Application to Quadruped Locomotion (https://blogs.cuit.columbia.edu/zp2130/hierarchical_apprenticeship_learning_with_application_to_quadruped_locomotion/)
- Policy Gradient Methods (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_methods/)
- Actor-Critic Algorithms for Hierarchical Markov Decision Processes (https://blogs.cuit.columbia.edu/zp2130/actor-critic_algorithms_for_hierarchical_markov_decision_processes/)
- Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation (https://blogs.cuit.columbia.edu/zp2130/hierarchical_deep_reinforcement_learning_integrating_temporal_abstraction_and_intrinsic_motivation/)