强化学习2022
第8节

涉及知识点：
基于神经网络的策略梯度、A3C、确定性梯度策略、深度确定性策略梯度、TRPO、PPO

深度策略梯度

课程大纲

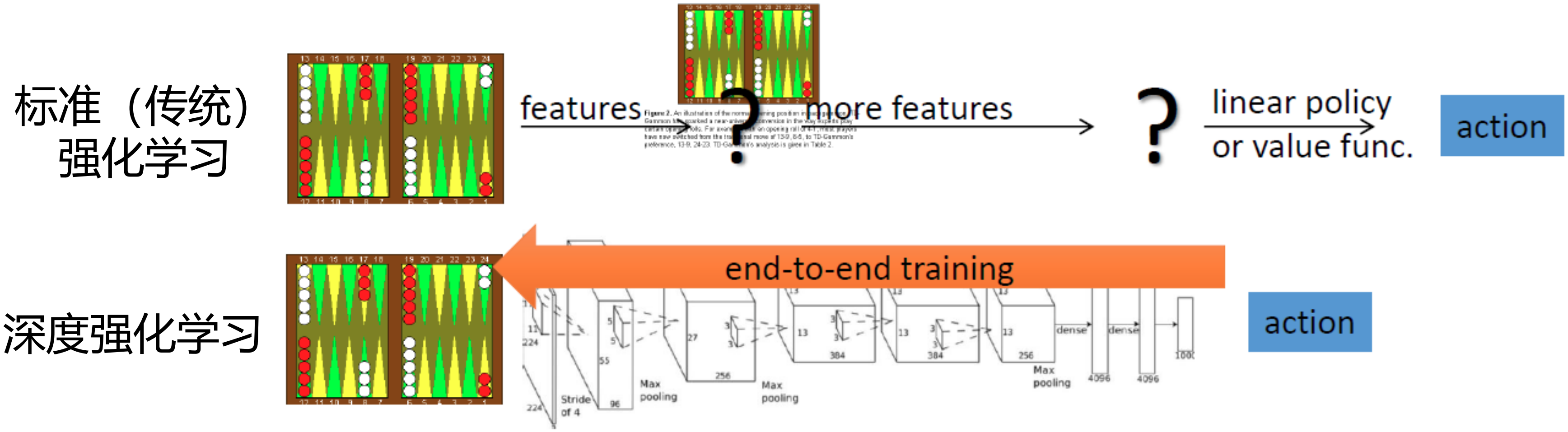
强化学习基础部分

1. 强化学习、探索与利用
2. MDP和动态规划
3. 值函数估计
4. 无模型控制方法
5. 规划与学习
6. 参数化的值函数和策略
7. 深度强化学习价值方法
8. 深度强化学习策略方法

强化学习前沿部分

9. 基于模型的深度强化学习
10. 离线强化学习
11. 模仿学习
12. 参数化动作空间
13. 多智能体强化学习基础
14. 多智能体强化学习前沿
15. 强化学习的应用
16. 技术交流与回顾

深度强化学习



- 直面理解：深度学习+强化学习
- 深度强化学习使强化学习算法能够以端到端的方式解决复杂问题
- 真正让强化学习有能力完成实际决策任务
- 比强化学习和深度学习各自都更加难以驯化
- 基于价值函数的深度强化学习
  - DQN：一次输入多个行动Q值输出、目标网络、随机采样经验
  - Double DQN：解耦合行动选择和价值估计、解决DQN过高估计问题
  - Dueling DQN：精细捕捉价值和行动的细微关联、多种advantage函数建模

深度强化学习的分类

- 基于价值的方法
  - 深度Q网络及其扩展
- 基于随机策略的方法
  - 基于神经网络的策略梯度，信任区域策略优化（TRPO），近端策略优化（PPO），A3C
- 基于确定性策略的方法
  - 确定性策略梯度（DPG），DDPG

基于神经网络的策略梯度

复习：策略梯度定理

- 策略梯度定理把似然比的推导过程泛化到多步马尔可夫决策过程
  - 用长期的价值函数 $Q^{\pi_\theta}(s,a)$代替前面的瞬时奖励 r_{sa}
- 策略梯度定理涉及
  - 起始状态目标函数J_1，平均奖励目标函数 J_{avR}，和平均价值目标函数 J_{avV}
- 定理
  - 对任意可微的策略 $\pi_\theta(a|s)$，任意策略的目标函数 $J = J_1, J_{avR}, J_{avV}$，其策略梯度是

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

详细证明过程请参考Rich Sutton’s Reinforcement Learning: An Introduction (2nd Edition)第13章

策略网络的梯度

- 对于随机策略，一般采样到每一个行动的概率由Softmax实现

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

  - 其中$f_\theta(s,a)$是对状态-行动队的打分函数，由θ参数化，这可以通过一个神经网络来实现

- 其log形式的梯度为

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \frac{1}{\sum_{a'} e^{f_\theta(s,a')}} \sum_{a''} e^{f_\theta(s,a'')} \frac{\partial f_\theta(s,a'')}{\partial \theta}$$

$$= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]$$

策略网络的梯度

- 其log梯度形式

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]$$

- 策略网络的梯度为

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

$$= \mathbb{E}_{\pi_\theta}\left[\left(\underbrace{\frac{\partial f_\theta(s,a)}{\partial \theta}}_{\text{反向梯度传播}} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\underbrace{\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]}_{\text{反向梯度传播}}\right) Q^{\pi_\theta}(s,a)\right]$$

策略梯度和Q学习的对比

- Q 学习算法学习一个由 θ 作为参数的函数 $Q_\theta(s,a)$
  - 优化目标为最小化TD error
$$J(\theta)=\mathbb{E}_{\pi'}\left[\frac{1}{2}\left(r_t+\gamma\max_{a'}Q_{\theta'}(s_{t+1},a')-Q_\theta(s_t,a_t)\right)^2\right]$$
  - 更新方程
$$\theta\leftarrow\theta-\alpha\frac{\partial J(\theta)}{\partial\theta}$$
$$=\theta+\alpha\mathbb{E}_{\pi'}\left[\left(r_t+\gamma\max_{a'}Q_{\theta'}(s_{t+1},a')-Q_\theta(s_t,a_t)\right)\frac{\partial Q_\theta(s,a)}{\partial\theta}\right]$$

- 策略梯度学习一个由 θ 作为参数的策略$\pi_\theta(a|s)$
  - 优化目标直接为策略的价值（比Q学习更加直接）
$$\max_\theta J(\theta)=\mathbb{E}_{\pi_\theta}[\pi_\theta(a|s)\,A^{\pi_\theta}(s,a)]$$
  - 更新方程
$$\theta\leftarrow\theta+\alpha\frac{\partial J(\theta)}{\partial\theta}=\theta+\alpha\mathbb{E}_{\pi_\theta}\left[\frac{\partial\log\pi_\theta(a|s)}{\partial\theta}A^{\pi_\theta}(s,a)\right]$$

A3C

复习：Actor-Critic

- Actor-Critic的思想
  - REINFORCE策略梯度方法：使用蒙特卡罗采样直接估计(s_t, a_t)的值G_t
  - 为什么不建立一个可训练的值函数Q_Φ来完成这个估计过程?
- 演员（Actor）和评论家（Critic）

演员 $\pi_\theta(a|s)$

采取动作使评论家满意的策略

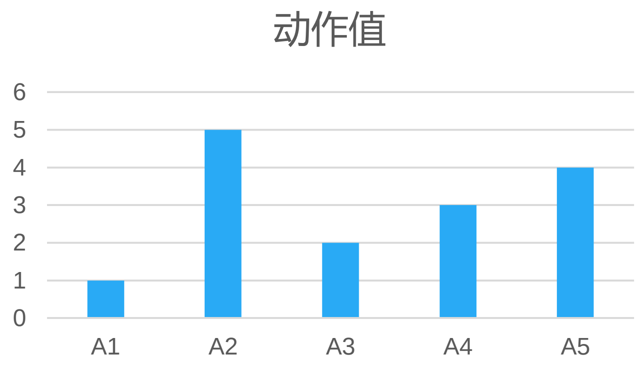
评论家 $Q_\Phi(s,a)$

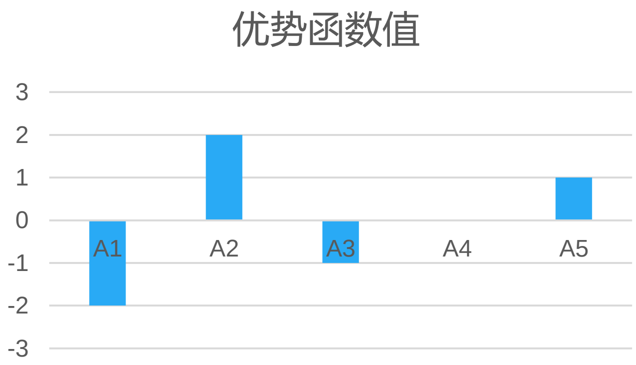
学会准确估计演员策略所采取动作价值的值函数

复习A2C：优势Actor-Critic

- 思想：通过减去一个基线函数来标准化评论家的打分
  - 更多信息指导：降低较差动作概率，提高较优动作概率
  - 进一步降低方差
- 优势函数（Advantage Function）

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$





A3C: 异步A2C方法

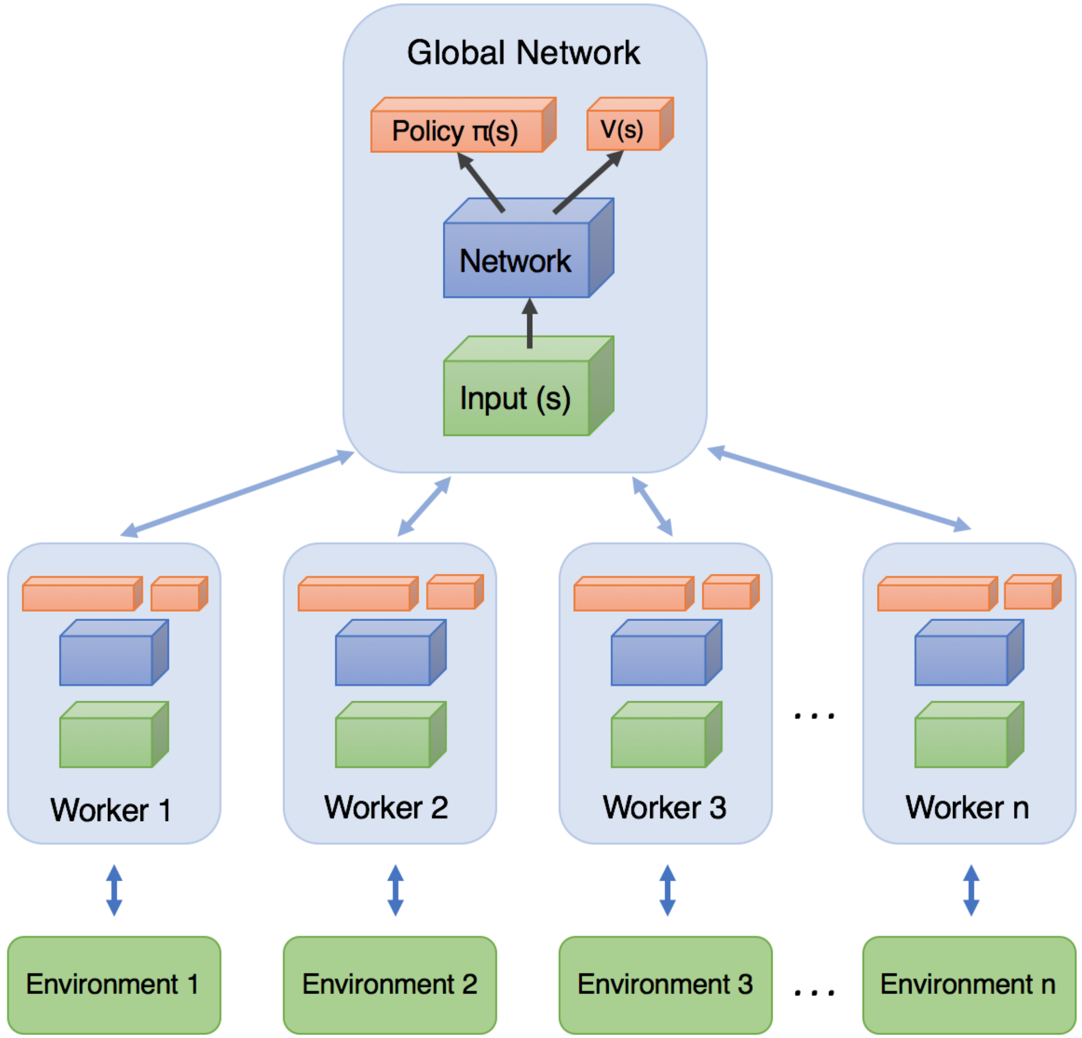
- A3C代表了异步优势动作评价（Asynchronous Advantage Actor Critic）
  - 异步（Asynchronous）：因为算法涉及并行执行一组环境
  - 优势（Advantage）：因为策略梯度的更新使用优势函数
  - 动作评价（Actor Critic）：因为这是一种动作评价（actor-critic）方法，它涉及一个在学得的状态值函数帮助下进行更新的策略

$$\nabla_{\theta'} \log \pi(a_t | s_t; \theta') A(s_t, a_t; \theta_v)$$

$$A(s_t, a_t; \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v)$$

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

A3C: 异步A2C方法



https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

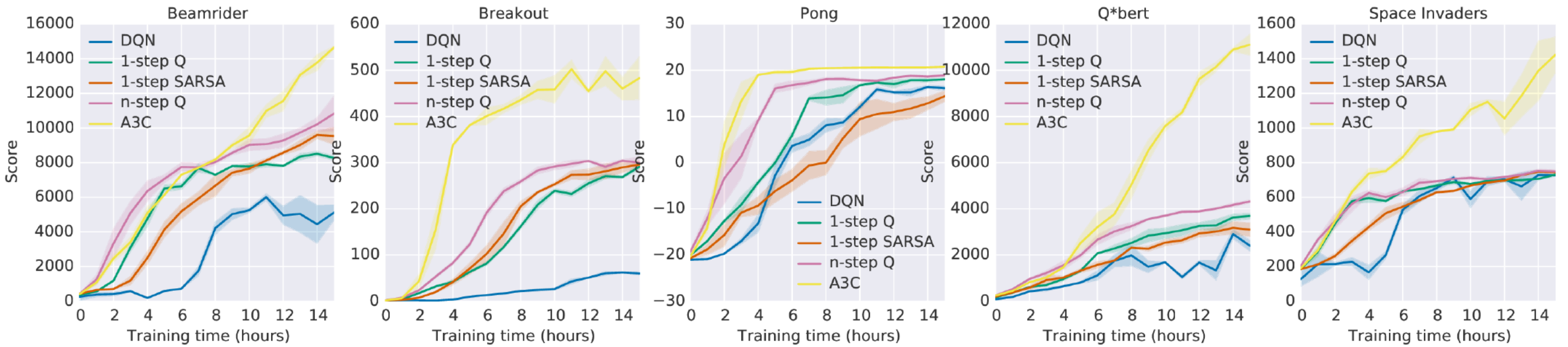
A3C算法

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Volodymyr Mnih et al. Asynchronous Methods for Deep Reinforcement Learning. NIPS 2016.

A3C对比实验



a single Nvidia K40 GPU while the asynchronous methods were trained using 16 CPU cores

Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

Nvidia K40 GPUs

16 CPU cores and no GPU

Mean and median human-normalized scores on 57 Atari games

Volodymyr Mnih et al. Asynchronous Methods for Deep Reinforcement Learning. NIPS 2016.

确定性策略梯度

深度强化学习的分类

- 基于价值的方法
  - 深度Q网络及其扩展
- 基于随机策略的方法
  - 使用神经网络的策略梯度，自然策略梯度，信任区域策略优化（TRPO），近端策略优化（PPO），A3C
- 基于确定性策略的方法
  - 确定性策略梯度（DPG），DDPG

随机策略与确定性策略

随机策略

对于离散动作

$$\pi(a|s;\theta) = \frac{\exp\{Q_\theta(s,a)\}}{\sum_{a'} \exp\{Q_\theta(s,a')\}}$$

对于连续动作

$$\pi(a|s;\theta) \propto \exp\left\{\left(a - \mu_\theta(s)\right)^2\right\}$$

确定性策略

对于离散动作

$$\pi(s;\theta) = \arg\max_a Q_\theta(s,a)$$

不可微

对于连续动作

$$a = \pi(s;\theta)$$

可微

确定性策略梯度

- 用于估计状态-动作值的评论家（critic）模块

$$Q^w(s,a) \simeq Q^\pi(s,a)$$

$$L(w) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[\left(Q^w(s,a) - Q^\pi(s,a)\right)^2\right]$$

- 确定性策略
  - 确定性策略梯度定理

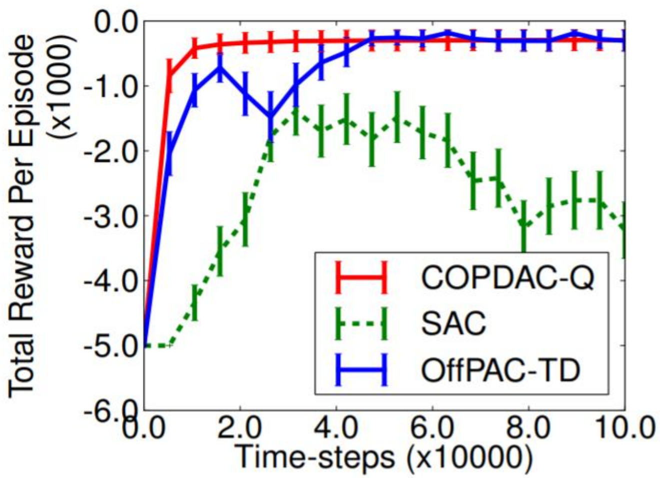
$$J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[Q^w(s,a)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \cdot \nabla_a Q^w(s,a)|_{a=\pi_\theta(s)}]$$

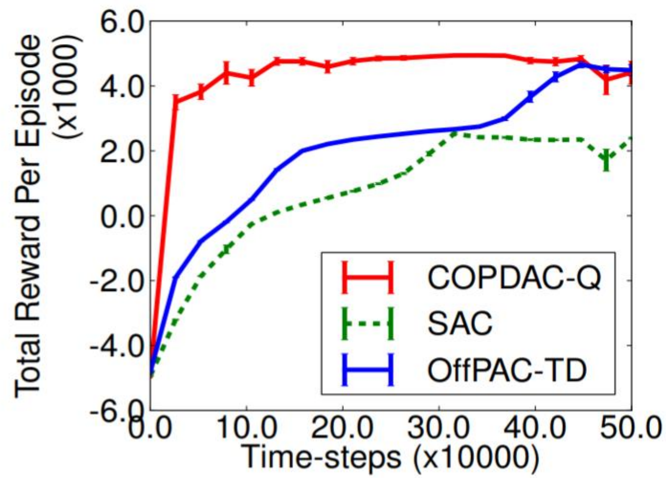
在线策略　　　　链式法则

D. Silver et al. Deterministic Policy Gradient Algorithms. ICML 2014.

确定性策略梯度实验效果

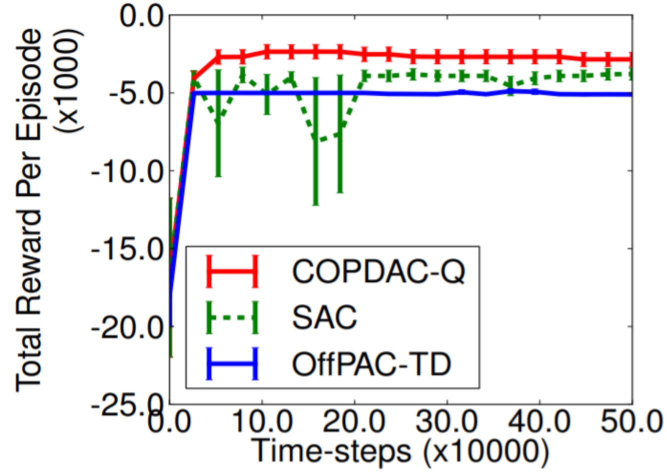


(a) Mountain Car



(b) Pendulum

- COPDAC-Q: 当时论文提出的确定性策略梯度（off-policy）
- SAC：随机梯度策略（on-policy）
- OffPAC-TD：随机梯度策略（off-policy）



(c) 2D Puddle World

D. Silver et al. Deterministic Policy Gradient Algorithms. ICML 2014.

深度确定性策略梯度

DDPG：深度确定性策略梯度

- 对于确定性策略的梯度

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \cdot \nabla_a Q^w(s,a)|_{a=\pi_\theta(s)}]$$

- 在实际应用中，这种带有神经函数近似器的actor-critic方法在面对有挑战性的问题时是不稳定的
- 深度确定性策略梯度（DDPG）给出了在确定性策略梯度（DPG）基础上的解决方法
  - 经验重放（离线策略）
  - 目标网络
  - 在动作输入前批标准化Q网络
  - 添加连续噪声

Lillicrap et al. Continuous control with deep reinforcement learning. ICLR 2016.

DDPG：深度确定性策略梯度

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**

动作上的噪声

    Select action $\boxed{a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t}$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions $\boxed{(s_i, a_i, r_i, s_{i+1}) \text{ from } R}$ 离线策略
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ 更新critic网络（a_i带有噪声）
    Update the actor policy using the sampled gradient:

目标critic网络

$$\nabla_{\theta^\mu} \mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

目标actor网络

更新actor网络

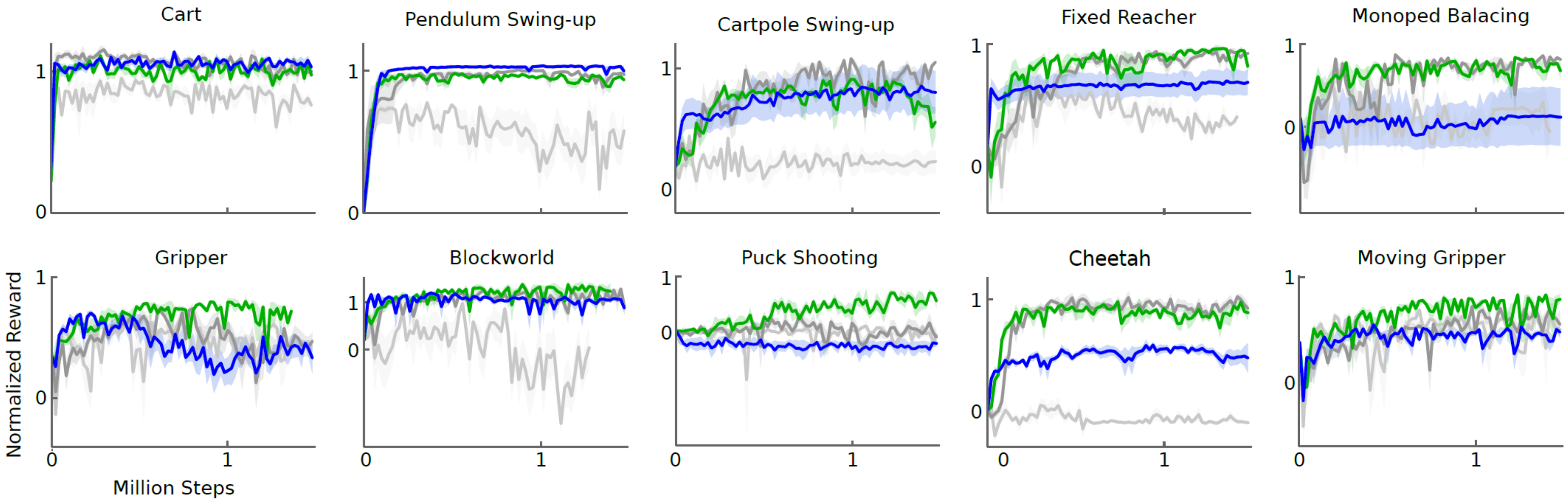
    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Lillicrap et al. Continuous control with deep reinforcement learning. ICLR 2016.

深度确定性策略梯度实验



- 确定性策略梯度（DPG）及其变种在一系列经典强化学习任务中的表现曲线
  - 浅灰色：使用批标准化的原始DPG算法
  - 暗灰色：使用目标网络的原始DPG算法
  - 绿色：同时使用目标网络和批标准化
  - 蓝色：使用仅像素作为输入的目标网络
- 目标网络至关重要

Lillicrap et al. Continuous control with deep reinforcement learning. ICLR 2016.

信任区域策略优化
TRPO

目录

Contents

01

策略梯度的缺点

策略梯度算法回顾

- 蒙特卡洛策略梯度（REINFORCE）算法

initialize θ arbitrarily

for each episode $\{s_1, a_1, r(s_1, a_1), \ldots, s_T, a_T, r(s_T, a_T)\} \sim \pi_\theta$ do

for $t = 1$ to T do

$$\theta \leftarrow \theta + \alpha \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|s_t) G_t$$

end for

end for

return θ

相关定义

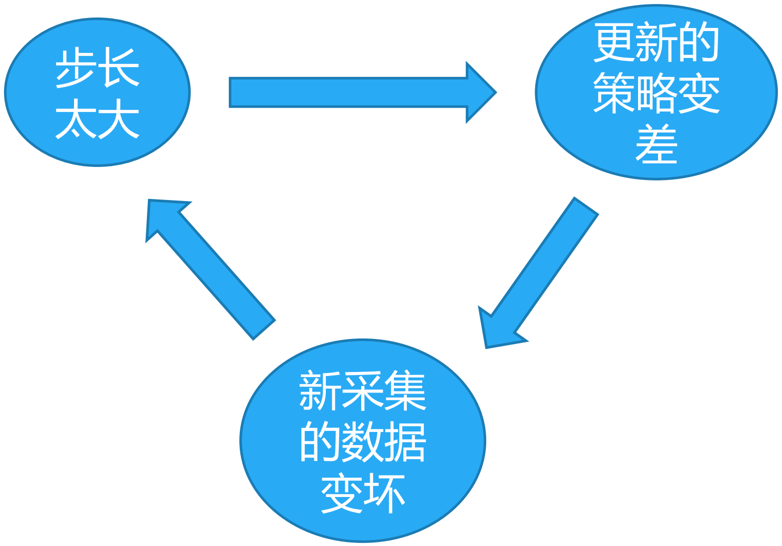
- s_t， a_t， $r(s_t, a_t)$： t 时刻的状态，动作和奖励
- π_θ， θ： 使用的策略，表示策略所使用的参数
- G_t： 累计奖励
- α： 步长

“OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS” ，John Schulman. (2016)

策略梯度的缺点

步长

- 步长难以确定
  - 采集到的数据的分布会随策略的更新而变化。
  - 较差的步长产生的影响大。



“OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS”，John Schulman. (2016)

02

TRPO

策略梯度的优化目标

- 优化目标的两种形式
  - 第一种：$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[\sum_t \gamma^t r(s_t, a_t)]$
  - 因为 $V^{\pi_\theta}(s) = \mathbb{E}_{a\sim\pi_\theta(s)}[Q^{\pi_\theta}(s,a)] = \mathbb{E}_{a\sim\pi_\theta(s)}\left[\mathbb{E}_{\tau\sim p_\theta(\tau)}[\sum_{s_k=s,a_k=a}\sum_{t=k}^{\infty}\gamma^{t-k}r(s_t,a_t)]\right]$
  - 所以优化目标的第二种形式是： $J(\theta) = \mathbb{E}_{s_0\sim p_\theta(s_0)}[V^{\pi_\theta}(s_0)]$

相关定义

- τ：轨迹
- s_0：初始状态
- s_t， a_t， $r(s_t, a_t)$： t 时刻的状态，动作和奖励
- π_θ：使用的策略
- θ：表示策略所使用的参数
- Q^{π_θ} 和 V^{π_θ}：策略 π_θ 下的 Q 值与状态值函数

“OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS”，John Schulman. (2016)

优化目标的优化量

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[\textstyle\sum_t \gamma^t r(s_t, a_t)]$$
$$J(\theta) = \mathbb{E}_{s_0\sim p_\theta(s_0)}[V^{\pi_\theta}(s_0)]$$

$$\begin{aligned}
J(\theta') - J(\theta) &= J(\theta') - \mathbb{E}_{s_0\sim p(s_0)}[\mathrm{V}^{\pi_\theta}(s_0)] \\
&= J(\theta') - \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}[\mathrm{V}^{\pi_\theta}(s_0)] \\
&= J(\theta') - \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t \mathrm{V}^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty}\gamma^t \mathrm{V}^{\pi_\theta}(s_t)\right] \\
&= J(\theta') + \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t (\gamma \mathrm{V}^{\pi_\theta}(s_{t+1}) - \mathrm{V}^{\pi_\theta}(s_t))\right] \\
&= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] + \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t (\gamma \mathrm{V}^{\pi_\theta}(s_{t+1}) - \mathrm{V}^{\pi_\theta}(s_t))\right] \\
&= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t (r(s_t, a_t) + \gamma \mathrm{V}^{\pi_\theta}(s_{t+1}) - \mathrm{V}^{\pi_\theta}(s_t))\right] \\
&= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_\theta}(s_t, a_t)\right]
\end{aligned}$$

初始状态的分布与θ无关

$J(\theta')$的定义

$Q^{\pi_\theta}(s_t, a_t)$

$A^{\pi_\theta}(s_t, a_t)$
$= Q^{\pi_\theta}(s_t, a_t) - \mathrm{V}^{\pi_\theta}(s_t)$

不方便采样

“OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS” , John Schulman. (2016)

使用重要性采样

- 使用重要性采样（Importance Sampling）

$$
\begin{aligned}
& J(\theta') - J(\theta) \\
&= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t)\right] \\
&= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}[\mathbb{E}_{a_t \sim \pi_{\theta'}(a_t|s_t)}[\gamma^t A^{\pi_\theta}(s_t, a_t)]] \\
&= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]
\end{aligned}
$$

$$
\begin{aligned}
& A^{\pi_\theta}(s_t, a_t) \\
&= Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)
\end{aligned}
$$

仍然是 $p_{\theta'}$
(近似操作)

重要性采样

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS"，John Schulman. (2016)

忽略状态分布的差异

- 当策略更新前后的变化较小时，可以令 $p_\theta(s_t) \approx p_{\theta'}(s_t)$。
  - 假设使用确定性策略，当 $\pi_{\theta'}(s_t) \neq \pi_\theta\ (s_t)$ 的概率小于ϵ时
  - 或者假设使用随机策略，当$a' \sim \pi_{\theta'}(\cdot\,|s_t) \neq a \sim\ \pi_\theta(\cdot\,|s_t)$的概率小于$\epsilon$时
  - $p_{\theta'}(s_t) = (1-\epsilon)^t p_\theta(s_t) + (1-(1-\epsilon)^t) p_{mistake}(s_t)$
  - $|p_{\theta'}(s_t) - p_\theta(s_t)| = (1-(1-\epsilon)^t)|p_{mistake}(s_t) - p_\theta(s_t)| \leq 2(1-(1-\epsilon)^t) \leq 2\epsilon t$

$$(1-\epsilon)^t \geq 1 - \epsilon t\ for\ \epsilon \in [0,1]$$

$$J(\theta') - J(\theta) \approx \sum_t \mathbb{E}_{s_t \sim p_{\theta}(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS"，John Schulman. (2016)

约束策略的变化

- 使用KL散度约束策略更新的幅度

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$$such\ that\ \mathbb{E}_{s_t \sim p(s_t)}[D_{KL}\left(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)\right)] \leq \epsilon$$

- 实际多使用constraint violate as penalty

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]] \\ -\lambda(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$$

1. 优化上式，更新θ'
2. 更新 $\lambda \leftarrow \lambda + \alpha(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$

“OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS”，John Schulman. (2016)
http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-9.pdf

Natural Policy Gradient

schemes. The natural policy gradient (Kakade, 2002) can be obtained as a special case of the update in Equation (12) by using a linear approximation to L and a quadratic approximation to the $\overline{D}_{\mathrm{KL}}$ constraint, resulting in the following problem:

$$\underset{\theta}{\text{maximize}} \left[\nabla_\theta L_{\theta_{\text{old}}}(\theta) \big|_{\theta=\theta_{\text{old}}} \cdot (\theta - \theta_{\text{old}}) \right] \tag{17}$$

$$\text{subject to } \frac{1}{2}(\theta_{\text{old}} - \theta)^T A(\theta_{\text{old}})(\theta_{\text{old}} - \theta) \leq \delta,$$

$$\text{where } A(\theta_{\text{old}})_{ij} =$$

$$\frac{\partial}{\partial \theta_i} \frac{\partial}{\partial \theta_j} \mathbb{E}_{s \sim \rho_\pi} \left[D_{\mathrm{KL}}(\pi(\cdot|s, \theta_{\text{old}}) \,\|\, \pi(\cdot|s, \theta)) \right] \big|_{\theta=\theta_{\text{old}}}.$$

The update is $\theta_{\text{new}} = \theta_{\text{old}} + \frac{1}{\lambda} A(\theta_{\text{old}})^{-1} \nabla_\theta L(\theta) \big|_{\theta=\theta_{\text{old}}}$,

TRPO的原理

Line search
(like gradient ascent)

Optimization in Trust Region

https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9

03

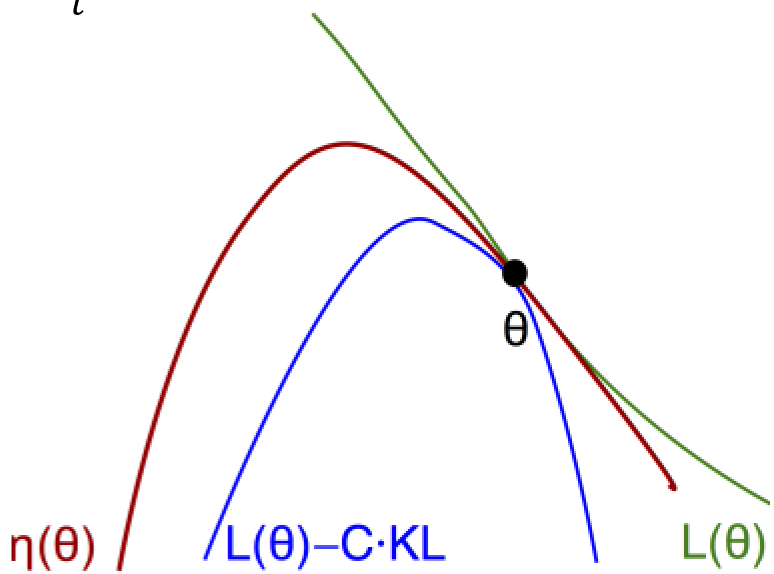
策略改进的单调性保证

策略改进的单调性保证

$$J(\theta') \geq L_\theta(\theta') - C \cdot D_{KL}^{max}(\theta, \theta'), where\ C = \frac{4\epsilon\gamma}{(1-\gamma)^2}, \epsilon = \max_{s,a} |A_\pi(s,a)|$$

$$L_\theta(\theta') = J(\theta) + \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$J(\theta')$

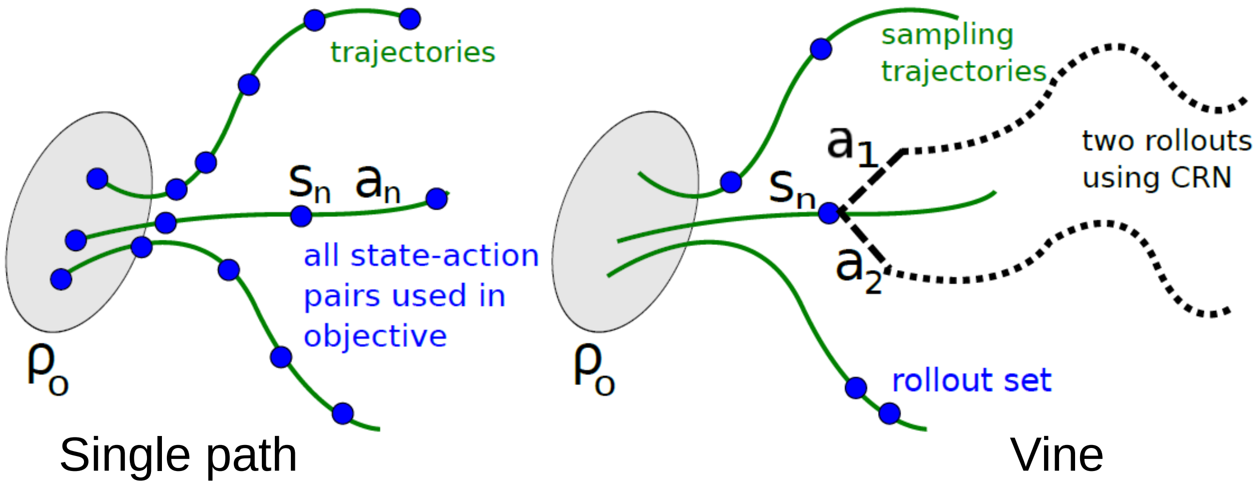


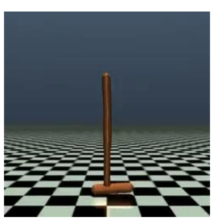
“Advanced Policy Gradient Methods: Natural Gradient, TRPO, and More ” , John Schulman. (2017)

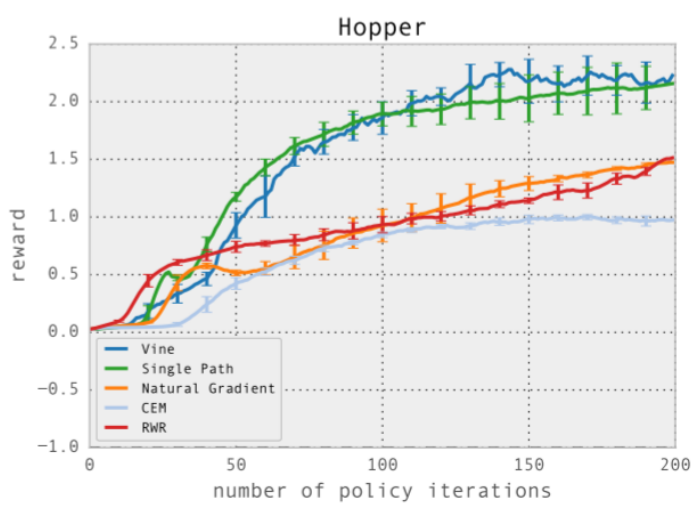
04

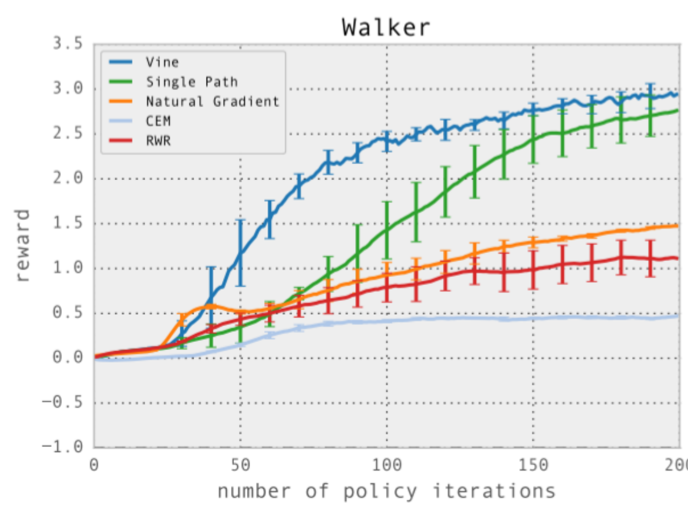
实验结果

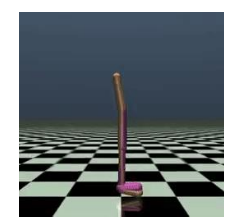
训练曲线











“Trust Region Policy Optimization”，John Schulman, et al. (2017)

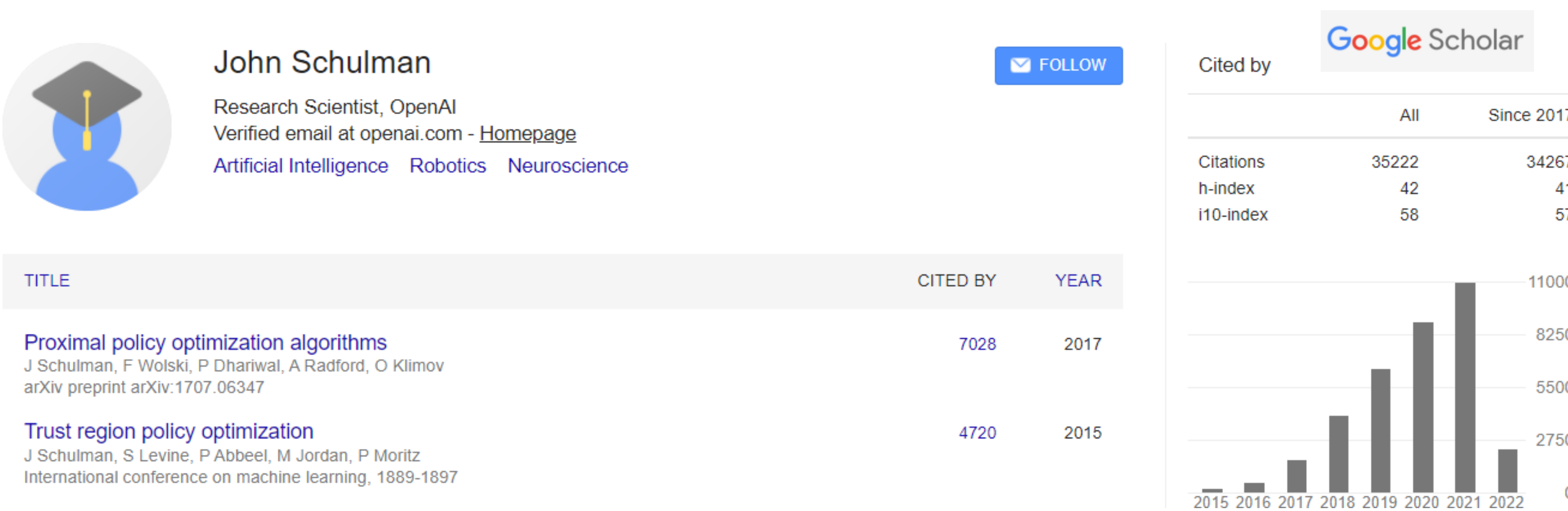
结果比较

	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	−20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	−3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2

推荐阅读

PPO

- TRPO的不足
  - 近似带来误差
  - 求解约束优化问题的困难
- PPO算法
  - 理论更简洁，操作更简单，实验效果更好
  - 推荐阅读 Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)



近端策略优化
Proximal Policy Optimization

回顾TRPO

- TRPO使用KL散度约束策略更新的幅度

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$$such\ that\ D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) \le \epsilon$$

  - 使用constraint violate as penalty

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]] - \lambda(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$$

1. 优化上式，更新θ'
2. 更新 $\lambda \leftarrow \lambda + \alpha(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$

TRPO的不足

- 重要性比例带来的大方差
- 求解约束优化问题的困难

PPO: Proximal Policy Optimization

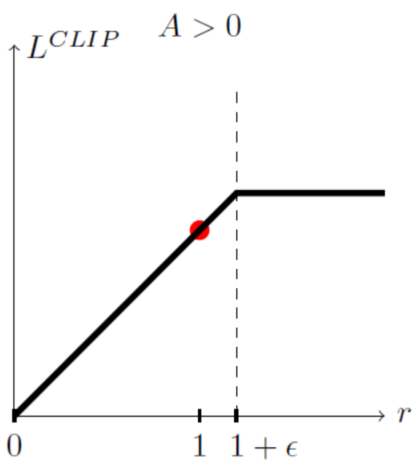
PPO在TRPO基础上的改进

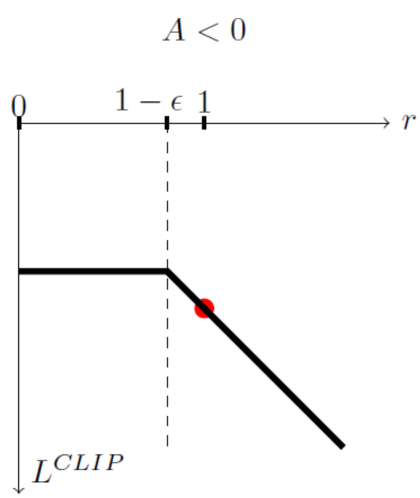
1. 截断式优化目标

conservative policy iteration

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$





构建下界

$$L^{CLIP}(\theta) \le L^{CPI}(\theta)$$

在$r = 1$附近相等

$$L^{CLIP}(\theta) = L^{CPI}(\theta)$$

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO: Proximal Policy Optimization

PPO在TRPO基础上的改进

1. 截断式优化目标

conservative policy iteration

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

2. 优势函数$\hat{A}_t$选用多步时序差分

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

- 在每次迭代中，并行N个actor收集T步经验数据
- 计算每步的$\hat{A}_t$ 和$L^{CLIP}(\theta)$，构成mini-batch
- 更新参数θ，并更新$\theta_{\text{old}} \leftarrow \theta$

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO: Proximal Policy Optimization

PPO在TRPO基础上的改进

3. 自适应的KL惩罚项参数

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t - \beta\ \mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot\,|s_t)|\pi_\theta(\cdot\,|s_t)]\right]$$

动态调整β方法

- 计算KL值 $d = \hat{\mathbb{E}}_t\left[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot\,|s_t)|\pi_\theta(\cdot\,|s_t)]\right]$
  a) 如果$d < d_{\text{targ}}/1.5$，更新$\beta \leftarrow \beta/2$
  b) 如果$d > d_{\text{targ}} \times 1.5$，更新$\beta \leftarrow \beta \times 2$

注：这里1.5和2是经验参数，算法效能和它们并不是很敏感

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO实验对比

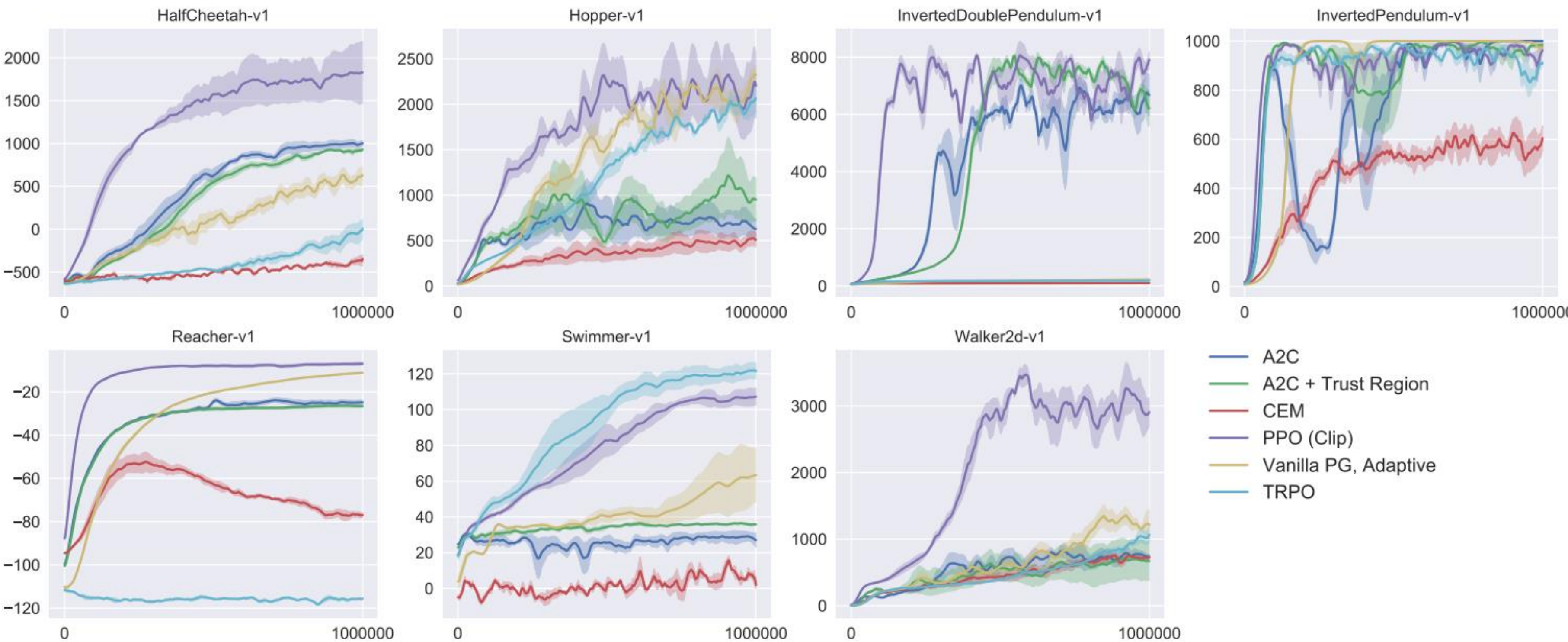
No clipping or penalty:	$L_t(\theta) = r_t(\theta)\hat{A}_t$
Clipping:	$L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$
KL penalty (fixed or adaptive)	$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}, \pi_\theta]$

- 7个连续控制的环境
- 3个random seed
- 每个算法跑100个episode，跑21遍，做平均值计算
- 最佳score归一化为1

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO实验对比



Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

总结深度策略梯度方法

- 相比价值函数学习最小化TD误差的目标，策略梯度方法直接优化策略价值的目标更加贴合强化学习本质目标
- 基于神经网络的策略在优化时容易因为一步走得太大而变得很差，进而下一轮产生很低质量的经验数据，进一步无法学习好
- Trust Region一类方法限制一步更新前后策略的差距（用KL散度），进而对策略价值做稳步地提升
- PPO在TRPO的基础上进一步通过限制importance ratio的range，构建优化目标的下界，进一步保证优化的稳定效果，是目前最常用的深度策略梯度算法
- 针对连续动作的决定性策略，可以从构建的critic中直接回传梯度到动作上，然后通过链式法则进一步将梯度回传到策略网络中
- 分布式的actor-critic算法能够充分利用多核CPU资源采样环境的经验数据，利用GPU资源异步地更新网络，这有效提升了DRL的训练效率

THANK YOU