强化学习2024
第7节

涉及知识点：
深度强化学习、深度Q网络、确定性策略梯度、
深度确定性策略梯度、双价值函数策略延时更新

深度强化学习

张伟楠 – 上海交通大学

课程回顾：基于表格的强化学习

基于模型的动态规划

- 值迭代 $V(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$
- 策略迭代 $\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V(s')$

无模型的强化学习

- 在线策略蒙特卡洛 $V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$
- 在线策略时序差分 $V(s_t) \leftarrow V(s_t) + \alpha\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$
- 在线策略时序差分 SARSA学习

  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
- 离线策略时序差分 Q-学习

  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

课程回顾：价值和策略的近似逼近方法

- 价值和策略的近似逼近方法是强化学习技术从‘玩具’ 走向 ‘现实’的第一步，是深度强化学习的基础设置
- 参数化的价值函数（$V_\phi(s), Q_\phi(s,a)$）和策略$\pi_\theta(a|s)$
- 通过链式法则，价值函数的参数可以被直接学习

$$\phi \leftarrow \phi + \alpha \left(r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s) \right) \frac{\partial V_\phi(s_t)}{\partial \phi}$$

无梯度

- 通过likelihood-ratio方法，可以用advantage对策略的参数进行学习

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

无梯度

- Actor-critic框架同时学习了价值函数和策略，通过价值函数的Q（或Advantage）估计，以策略梯度的方式更新策略参数

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

2024年上海交通大学ACM班强化学习课程大纲

强化学习基础部分

（中文课件）

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. 规划与学习
8. 深度强化学习价值方法
9. 深度强化学习策略方法

强化学习前沿部分

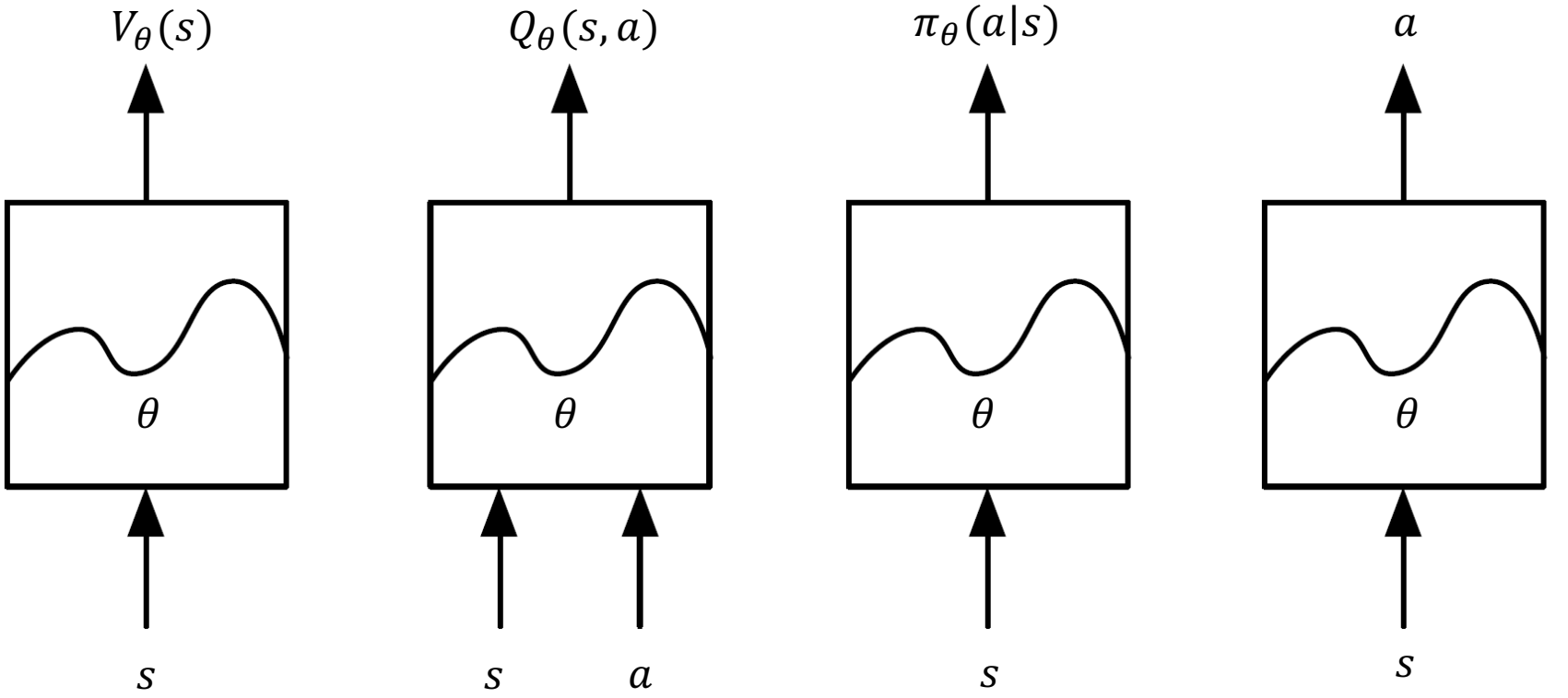
（英文课件）

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. 多智能体强化学习基础
13. 多智能体强化学习前沿
14. 基于扩散模型的强化学习
15. AI Agent与决策大模型
16. 技术交流与回顾

深度强化学习

讲师：张伟楠 – 上海交通大学

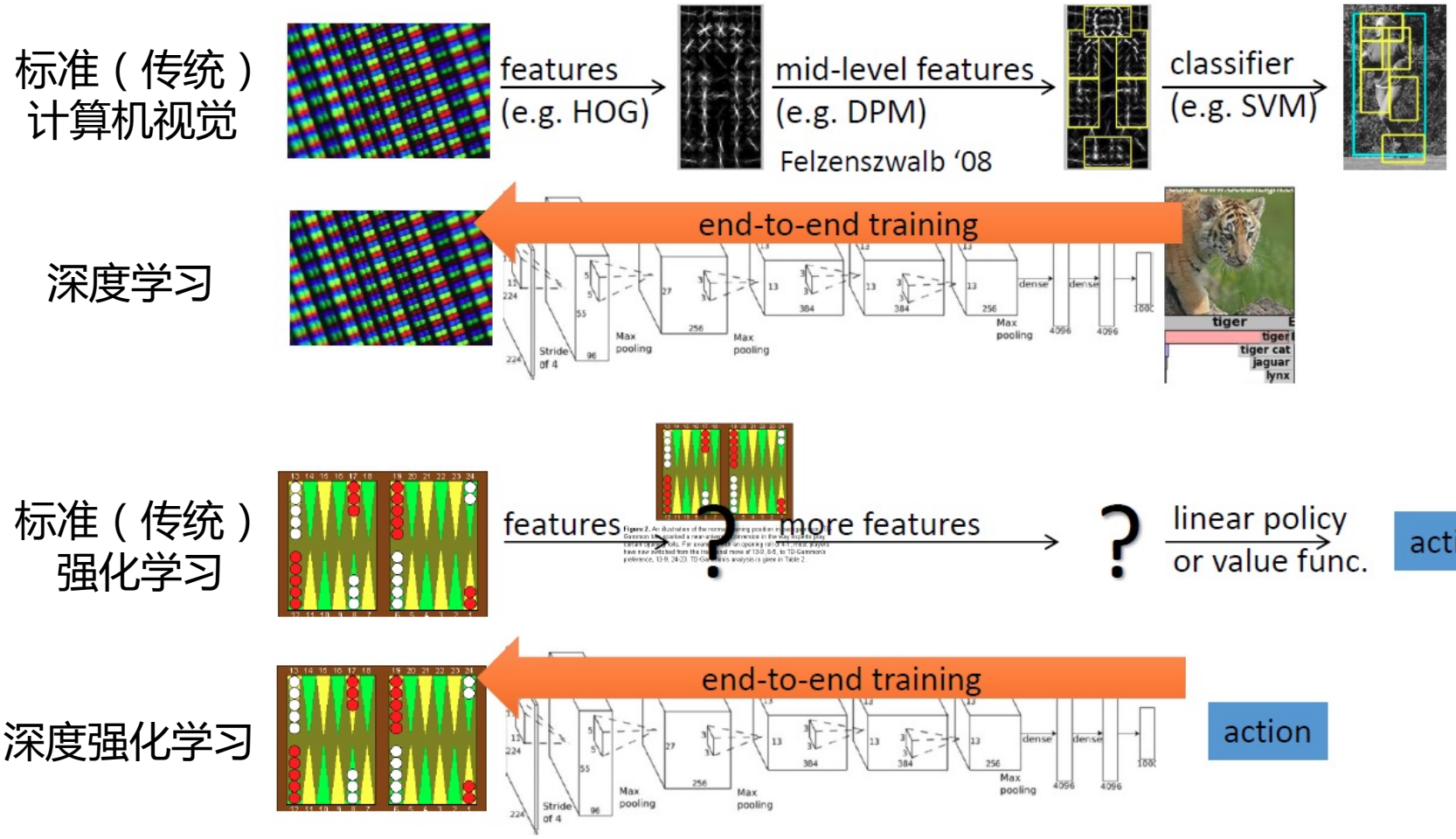
价值和策略近似



状态值函数和状态动作值函数近似　　随机策略近似　　确定性策略近似

- 假如我们直接使用深度神经网络建立这些近似函数呢？

端到端强化学习

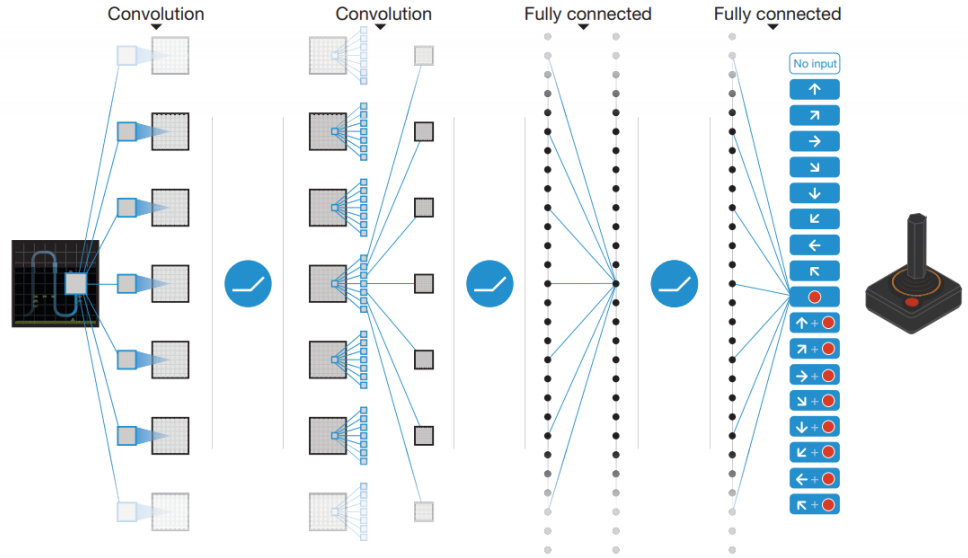


深度强化学习使强化学习算法能够以端到端的方式解决复杂问题

Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

深度强化学习

- 深度强化学习
  - 利用深度神经网络进行价值函数和策略近似
  - 从而使强化学习算法能够以端到端的方式解决复杂问题



Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度强化学习带来的关键变化

- 假如将深度学习（DL）和强化学习（RL）结合在一起会发生什么？
  - 价值函数和策略现在变成了深度神经网络
  - 相当高维的参数空间
  - 难以稳定地训练
  - 容易过拟合
  - 需要大量的数据
  - 需要高性能计算
  - CPU（用于收集经验数据）和GPU（用于训练神经网络）之间的平衡
  - …

这些新的问题促进着深度强化学习算法的创新

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
- 基于确定性策略的方法
  - 确定性策略梯度（DPG），DDPG，TD3
- 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

深度Q网络

讲师：张伟楠 – 上海交通大学

目录

Contents

01
深度Q网络（DQN）

Q 学习回顾

- 不直接更新策略
- 基于值的方法
- Q 学习算法学习一个由 θ 作为参数的函数 $Q_\theta(s,a)$

  - Target值 $y_t = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$
  - 更新方程 $Q_\theta(s_t,a_t) \leftarrow Q_\theta(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q_\theta(s_{t+1},a') - Q_\theta(s_t,a_t))$
  - 优化目标

$$\theta^\star \leftarrow \arg\min_\theta \frac{1}{2} \sum_{(s_t,a_t)\in D} (Q_\theta(s_t,a_t) - (r + \gamma \max_{a'} Q_\theta(s_{t+1},a'))^2$$

此处无梯度

深度Q网络（DQN）

直观想法

- 使用神经网络来逼近上述 $Q_\theta(s,a)$
  - 算法不稳定
    - 连续采样得到的 $\{(s_t,a_t,s_{t+1},r_t)\}$ 不满足独立分布。
    - $\{(s_t,a_t,s_{t+1},r_t)\}$ 为状态-动作-下一状态-回报输入。
    - $Q_\theta(s,a)$ 的频繁更新。

解决办法

- 经验回放
- 使用双网络结构：评估网络（evaluation network）和目标网络（target network）

“Human-Level Control Through Deep Reinforcement Learning”, Mnih, Kavukcuoglu, Silver et al. (2015)

经验回放

- 经验回放
  - 存储训练过程中的每一步 $e_t = (s_t, a_t, s_{t+1}, r_t)$ 于数据库 D 中，采样时服从均匀分布。

优先经验回放

- 衡量标准
  - 以 Q 函数的值与 Target 值的差异来衡量学习的价值，即

$$p_t = |r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a') - Q_\theta(s_t, a_t)|$$

  - 为了使各样本都有机会被采样，存储 $e_t = (s_t, a_t, s_{t+1}, r_t, p_t + \epsilon)$。
- 选中的概率
  - 样本 e_t 被选中的概率为 $P(t) = \frac{p_t^\alpha}{\sum_{k=1}^{N} p_k^\alpha}$
- 重要性采样（Importance Sampling）
  - 权重为 $\omega_j = \frac{(N \times P(j))^{-\beta}}{\max_i \omega_i}$

"Prioritized Experience Replay"，Schaul et al. (2016)

经验回放

Algorithm 1 Double DQN with proportional prioritization

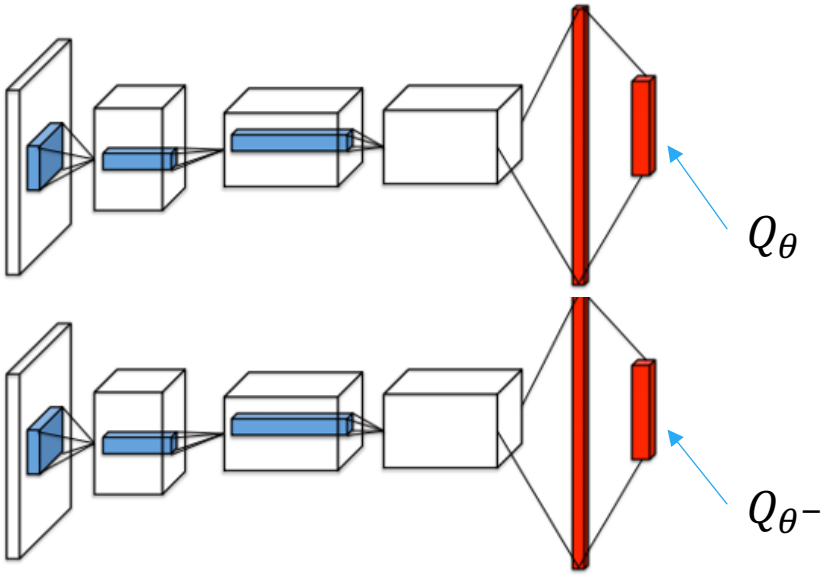
Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

"Prioritized Experience Replay"，Schaul et al. (2016)

目标网络

- 目标网络 $Q_{\theta^-}(s,a)$
  - 使用较旧的参数，记为 θ^-，每隔 C 步和训练网络的参数同步一次。
  - 第i次迭代的损失函数为

$$L_i(\theta_i) = \mathbb{E}_{s_t,a_t,s_{t+1},r_t,p_t\sim D}\left[\frac{1}{2}\omega_t(r_t + \gamma \max_{a'} Q_{\theta_i^-}(s_{t+1},a') - Q_{\theta_i}(s_t,a_t))^2\right]$$



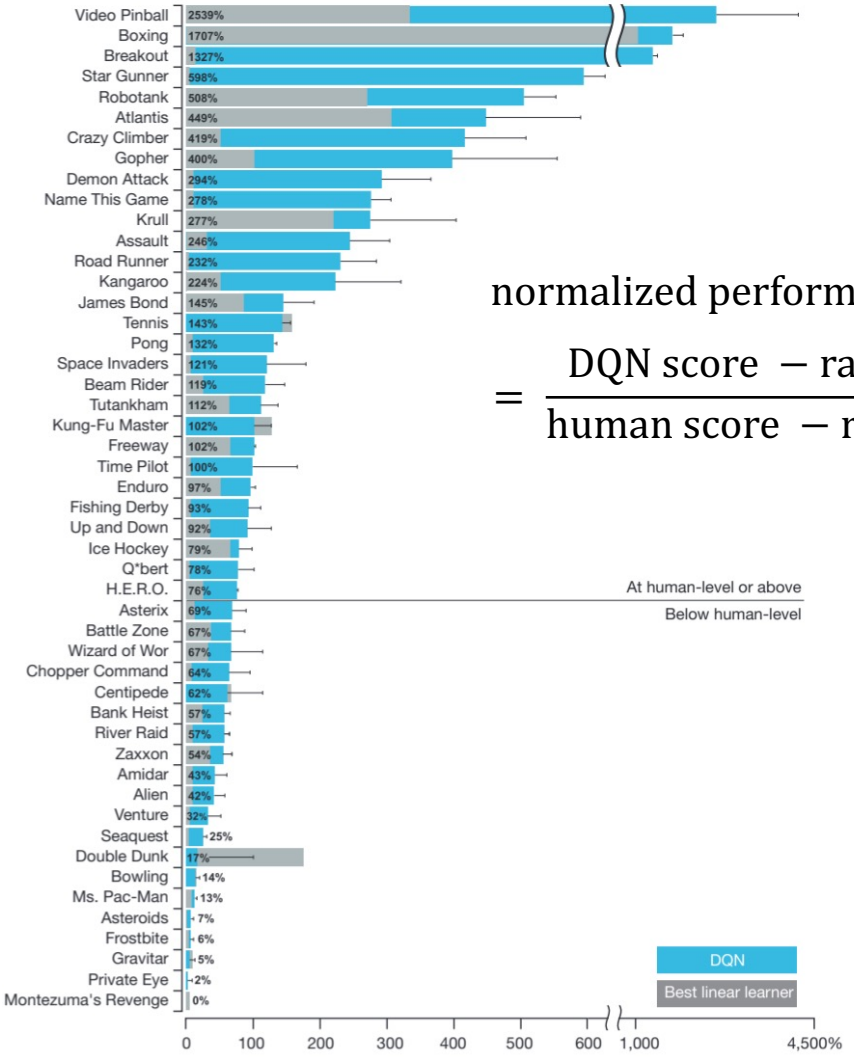
"Human-Level Control Through Deep Reinforcement Learning"，Mnih, Kavukcuoglu, Silver et al. (2015)

算法流程

1. 收集数据：使用 $\epsilon-$greedy 策略进行探索，将得到的状态动作组 (s_t, a_t, s_{t+1}, r_t) 放入经验池（replay-buffer）
2. 采样：从数据库中采样 k 个动作状态组
3. 更新网络
   - 用采样得到的数据计算 $Loss$
   - 更新 Q 函数网络 θ
   - 每 C 次迭代（更新Q函数网络）更新一次目标网络 θ^-

"Human-Level Control Through Deep Reinforcement Learning"，Mnih, Kavukcuoglu, Silver et al. (2015)

在 Atari 环境中的实验结果



$$\text{normalized performance} = \frac{\text{DQN score} - \text{random play score}}{\text{human score} - \text{random play score}}$$

The performance of DQN is normalized with respect to a professional human games tester (that is, 100% level)

"Human-Level Control Through Deep Reinforcement Learning", Mnih, Kavukcuoglu, Silver et al. (2015)

02
Double DQN

Q-learning中的过高估计

- Q函数的过高估计
  - Target值 $\mathrm{y_t} = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$

max 操作使得 Q 函数的值越来越大，甚至高于真实值

- 过高估计的原因

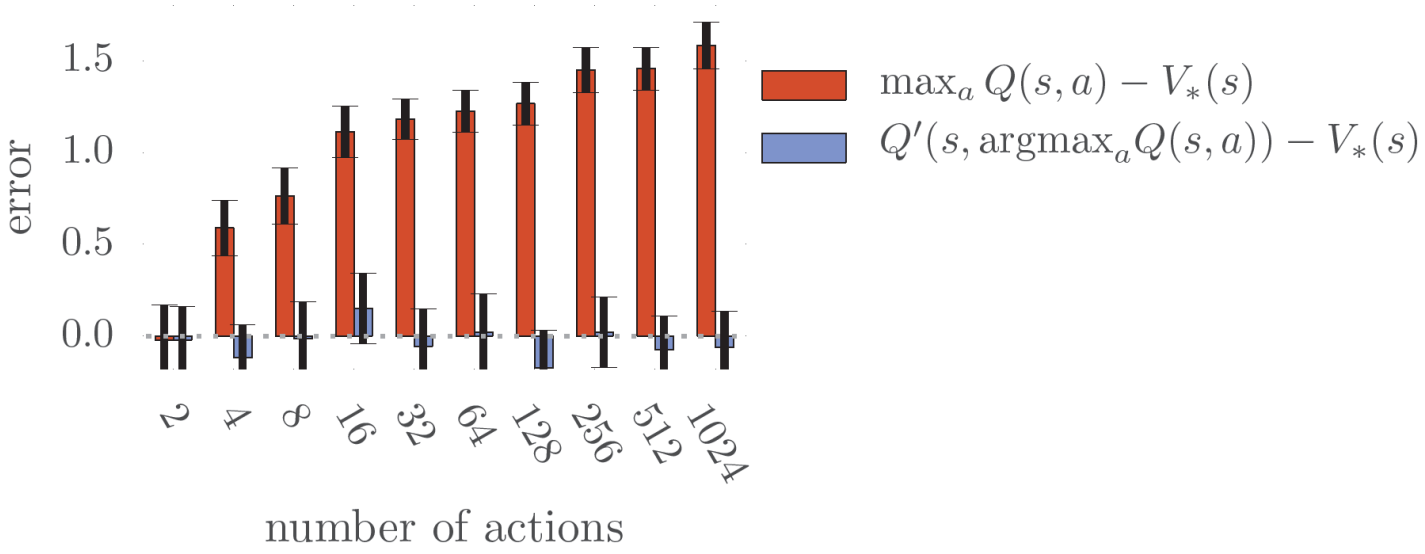
$$\max_{a' \in A} Q_{\theta'}(s_{t+1}, a') = Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_{\theta'}(s_{t+1}, a'))$$

此处选择的a'可能是由于Q函数错误地过高估计导致

"Double Reinforcement Learning with Double Q-Learning"，van Hasselt et al. (2016)
"Double Q-learning"，van Hasselt.(2010)

Q-learning中的过高估计

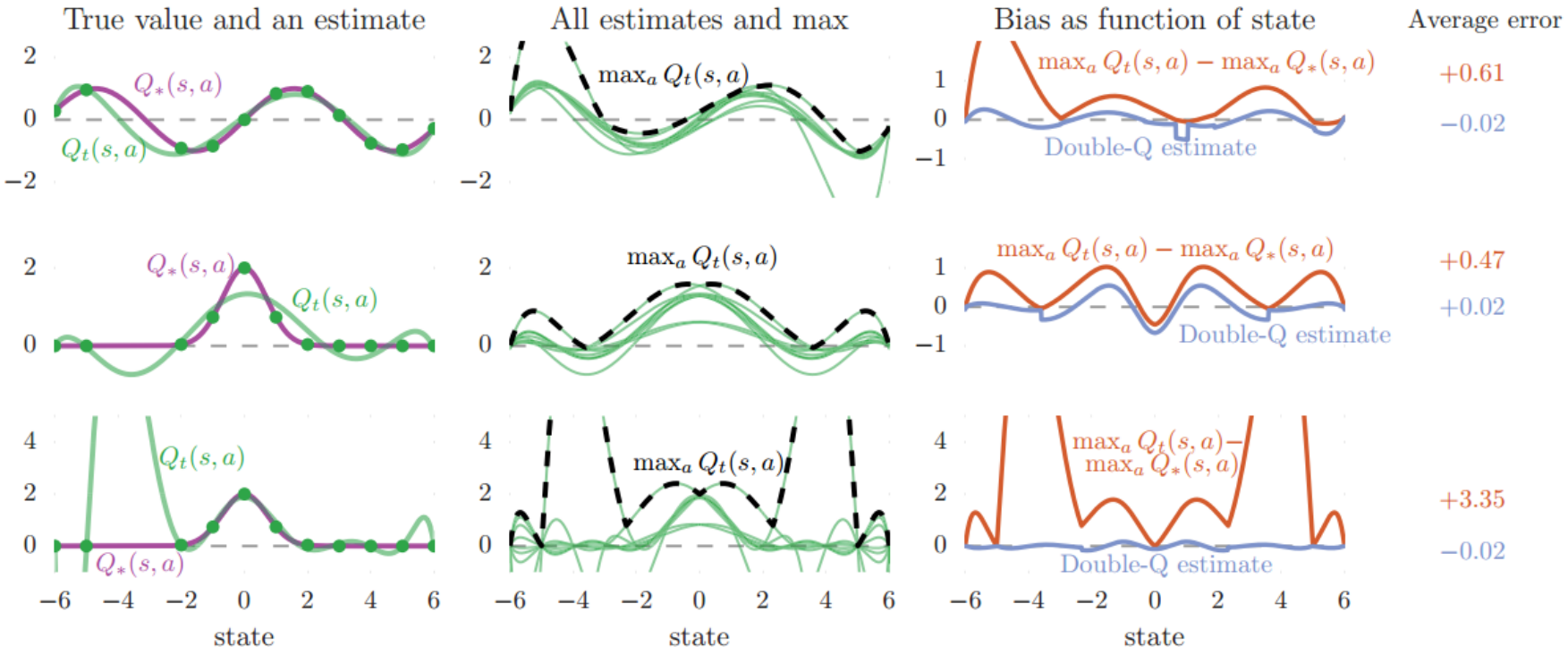
- Q函数的过高估计程度随着候选行动数量增大变得更严重



- 其中$Q_t(s,a) - V_*(s)$设为在[-1,1]区间均匀分布
- Q'函数是另一组独立训练的价值函数

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)
"Double Q-learning", van Hasselt.(2010)

Q-learning中过估计的例子



- 设置：x轴为状态，10个候选行动；紫线是真实价值函数，绿点是训练数据点，绿线是拟合的价值函数
- 中间列展示10个行动的$Q_t(s,a)$估计，在取max后，与真实$Q_*(s,a)$差距太大

"Double Reinforcement Learning with Double Q-Learning"，van Hasselt et al. (2016)

Double DQN

- 使用不同的网络来估值和决策

DQN $y_t = r_t + \gamma Q_\theta(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$

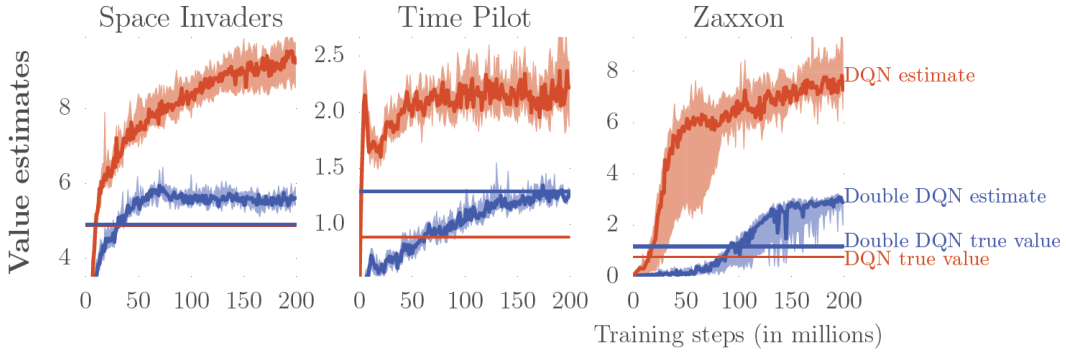
Double DQN $y_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$

裁判　　运动员

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

在 Atari 环境中的实验结果

- 价值估计误差

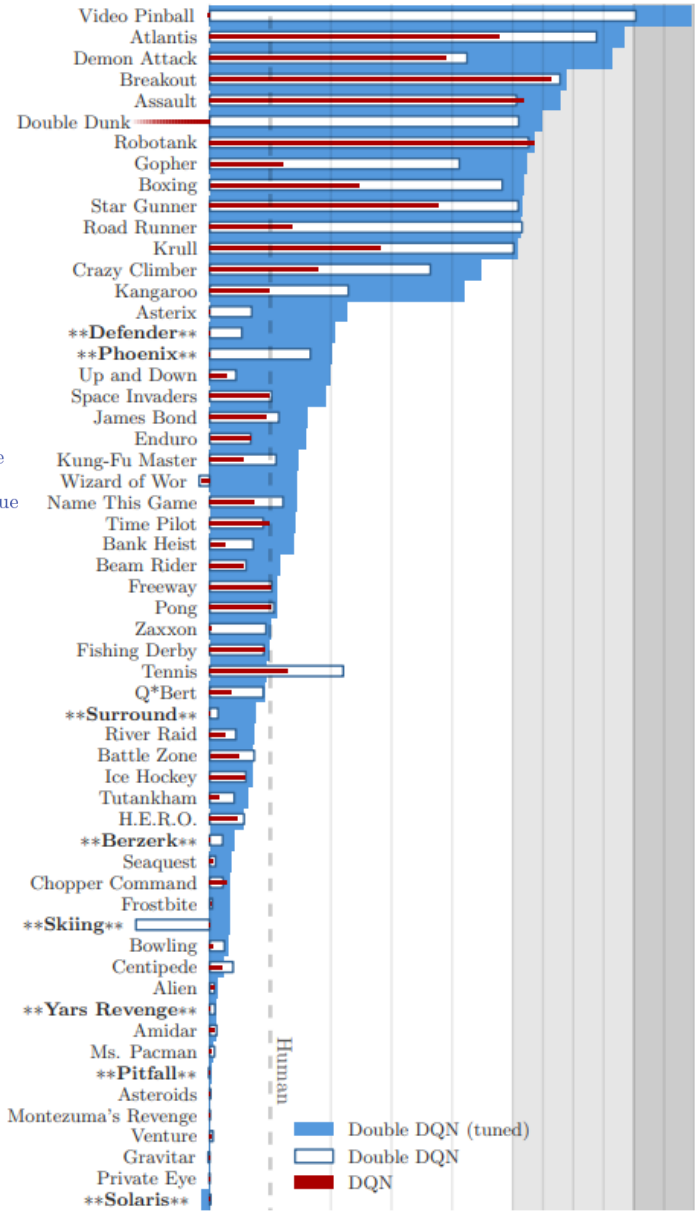


- Atari Performance

	no ops		human starts		
	DQN	DDQN	DQN	DDQN	DDQN (tuned)
Median	93%	**115%**	47%	88%	**117%**
Mean	241%	**330%**	122%	273%	**475%**

normalized performance

$$= \frac{\text{DQN score} - \text{random play score}}{\text{human score} - \text{random play score}}$$



"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

04
Dueling DQN

Dueling DQN

假设动作值函数服从某个分布：

$$Q(s,a) \sim \mathcal{N}(\mu,\sigma)$$

显然：$V(s) = \mathbb{E}[Q(s,a)] = \mu$

同样有：$Q(s,a) = \mu + \varepsilon(s,a)$

偏移量

问题

如何描述 $\varepsilon(s,a)$? → $\varepsilon(s,a) = Q(s,a) - V(s)$ → 也称为Advantage函数

Dueling DQN

Advantage 函数

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

$$Q^{\pi}(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$$

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(s)}[Q^{\pi}(s,a)]$$

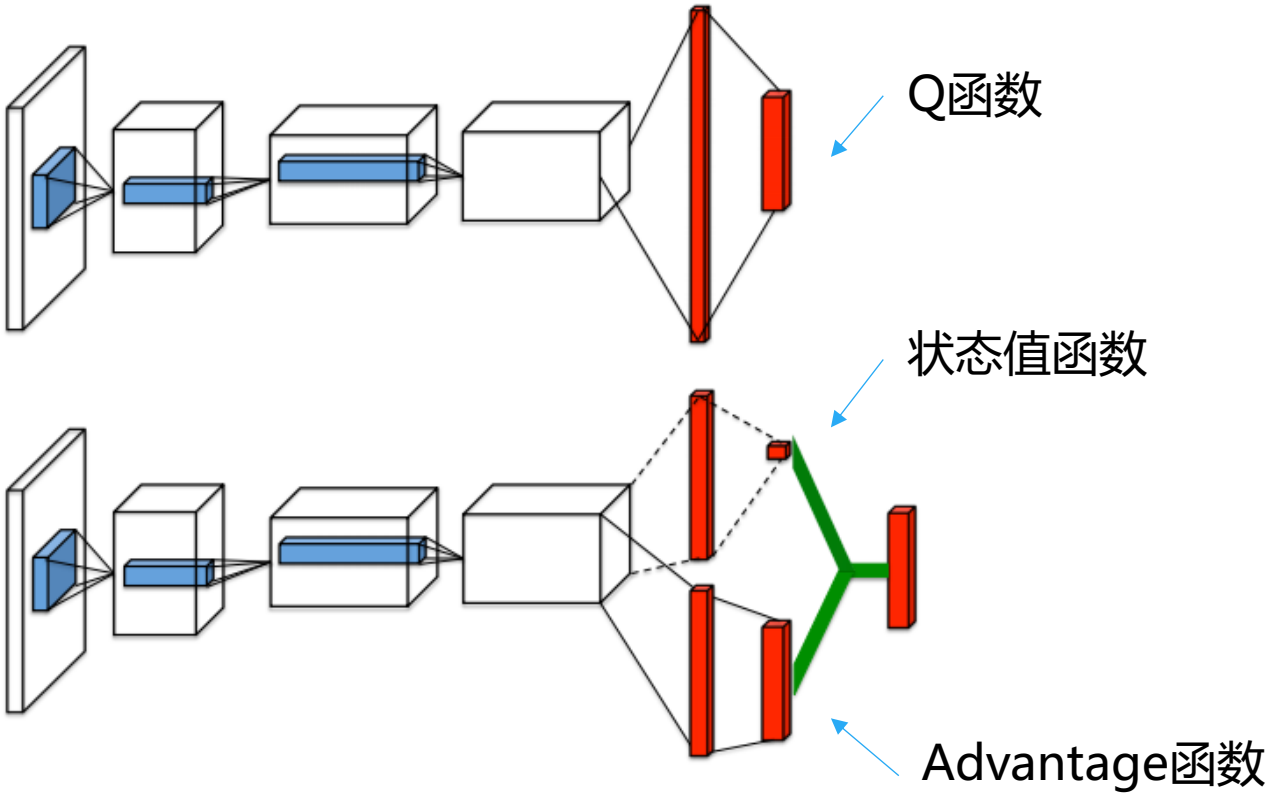
不同的Advantage聚合形式

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a'\in|A|} A(s,a';\theta,\alpha)\right)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

"Dueling Network Architectures for Deep Reinforcement Learning", Wang et al. (2016)

网络结构

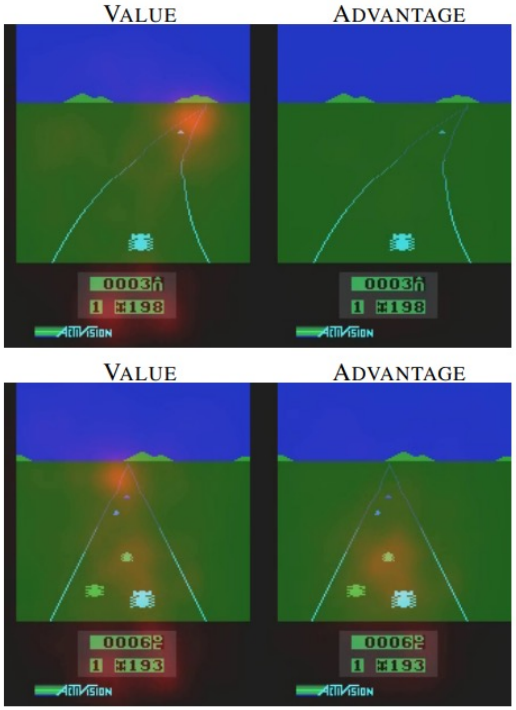


"Dueling Network Architectures for Deep Reinforcement Learning", Wang et al. (2016)

优点

- 处理与动作关联较小的状态
- 状态值函数的学习较为有效：一个状态值函数对应多个Advantage函数

显著区域
(saliency maps)

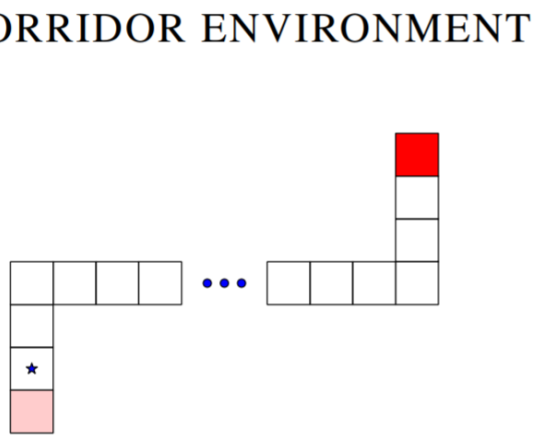


可以在不考虑动作的影响下判断出该状态的好坏。

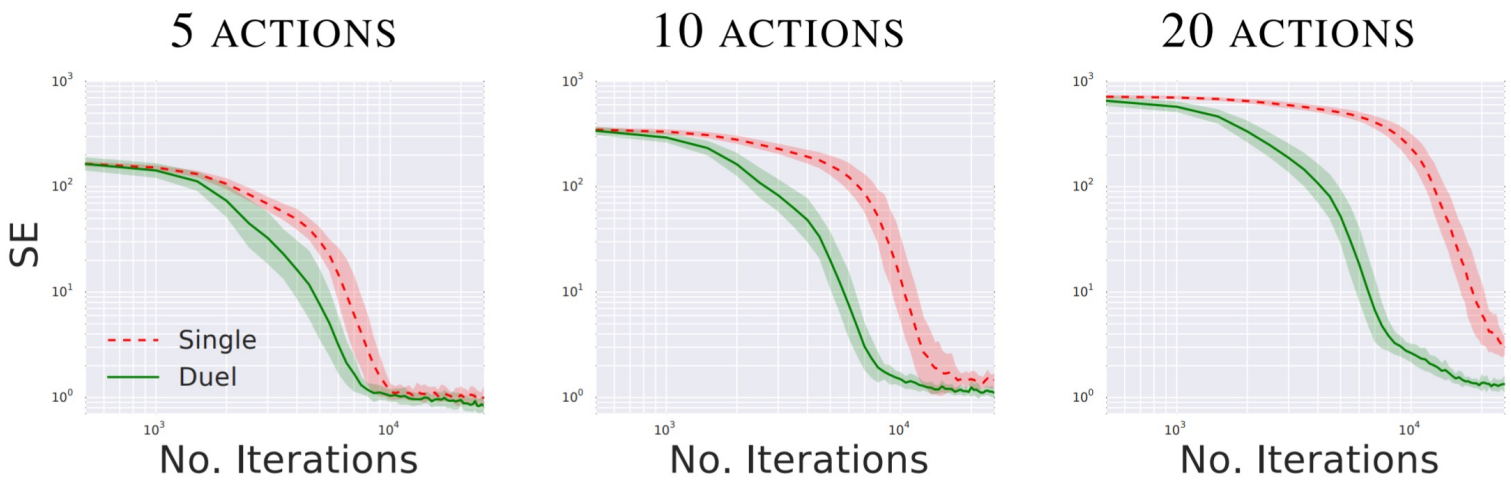
可以强调动作的重要性：advantage学会只在agent面前有车的时候会加强注意力

探索任务：走廊环境

- 走廊环境
  - *为起点状态
  - 行动：上、下、左、右、不动
  - 左下角有小的正向奖励
  - 右上角有大的正向奖励
- Q函数评估均方误差

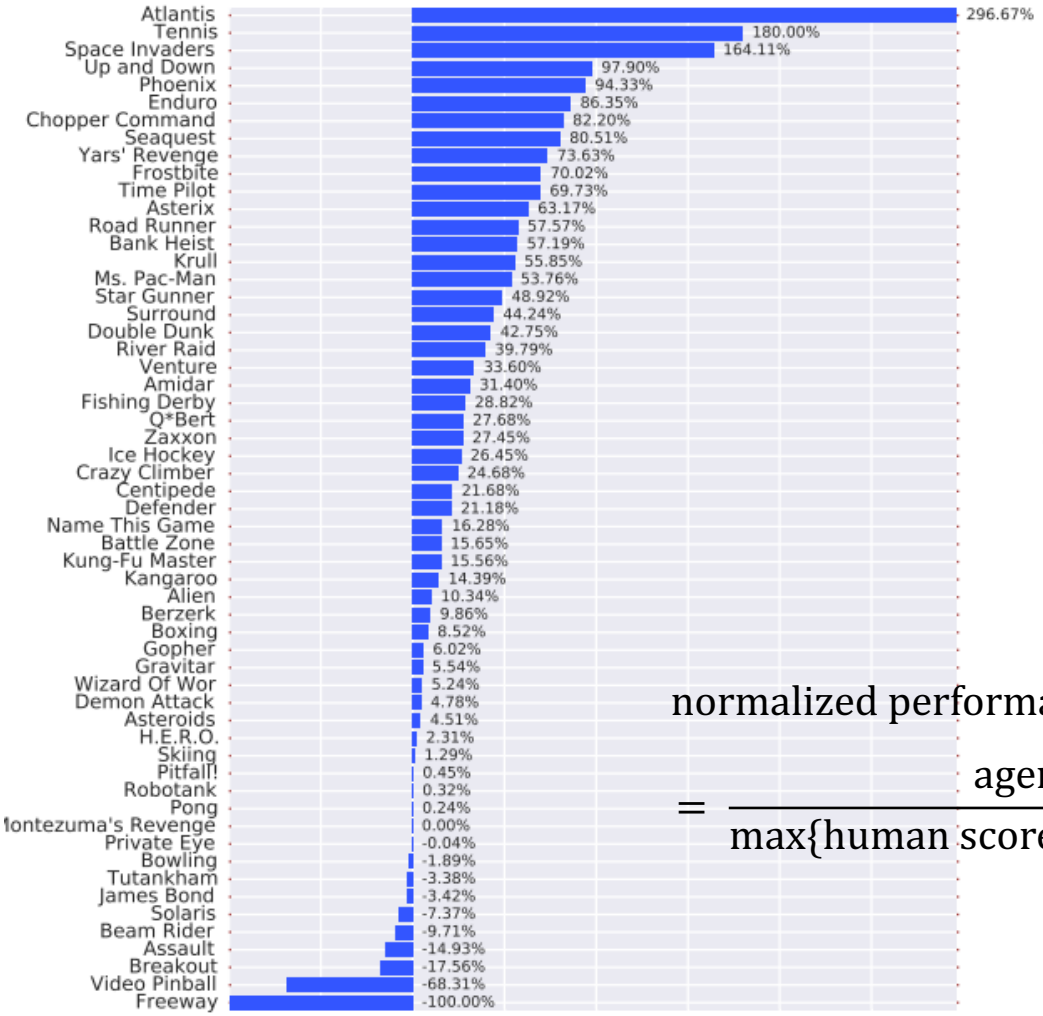


更多的行动是‘不动’



“Dueling Network Architectures for Deep Reinforcement Learning”, Wang et al. (2016)

在 Atari 环境中的实验结果I

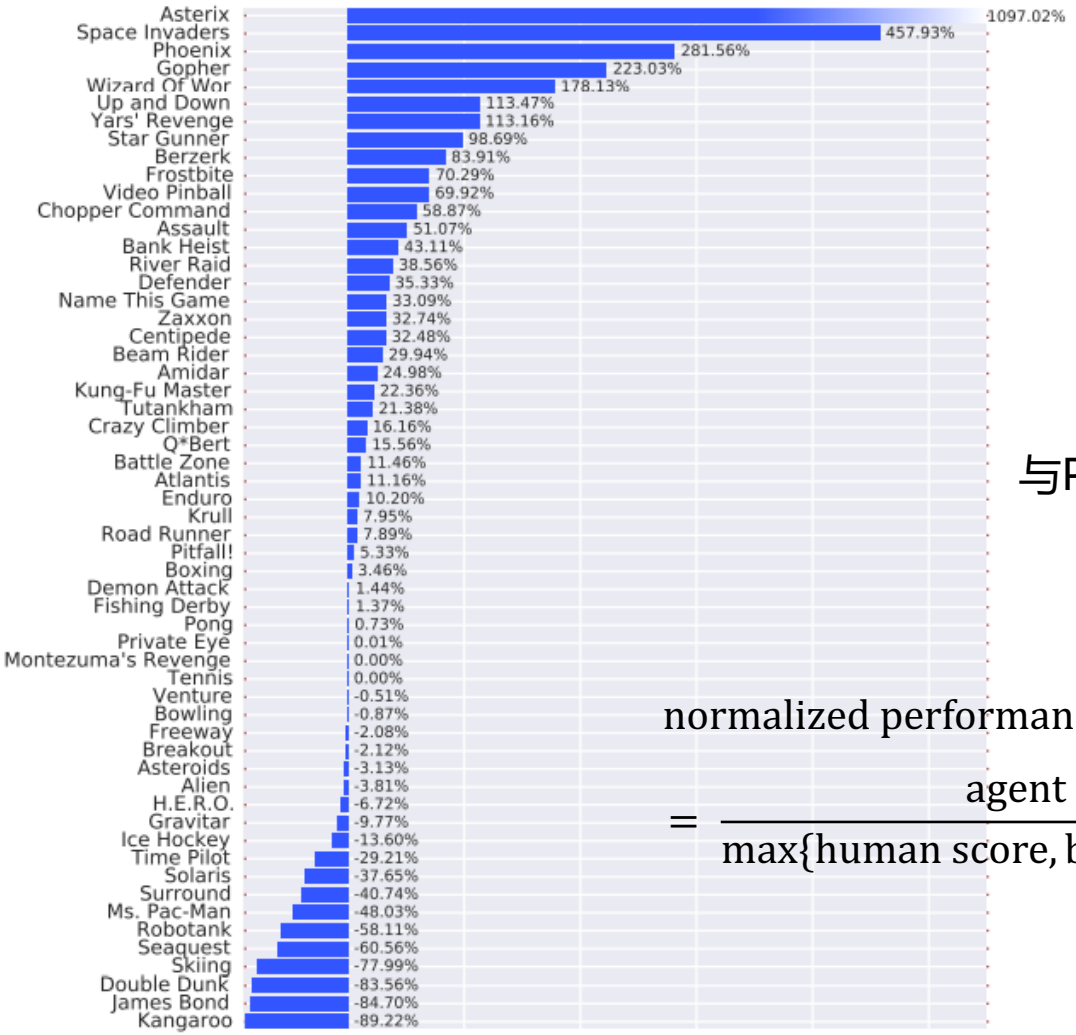


与DQN对比

$$\text{normalized performance} = \frac{\text{agent score} - \text{baseline score}}{\max\{\text{human score}, \text{baseline score}\} - \text{random play score}}$$

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

在 Atari 环境中的实验结果II



与Prioritized Double DQN对比

$$\text{normalized performance} = \frac{\text{agent score} - \text{baseline score}}{\max\{\text{human score}, \text{baseline score}\} - \text{random play score}}$$

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

复习之前的深度强化学习算法

- DQN：一次输入多个行动Q值输出、目标网络、随机采样经验

$$y_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_{\theta'}(s_{t+1}, a'))$$

“Human-Level Control Through Deep Reinforcement Learning”, Mnih, Kavukcuoglu, Silver et al. Nature 2015.

- Double DQN：解耦合行动选择和价值估计、解决DQN过高估计问题

$$y_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_{\theta}(s_{t+1}, a'))$$

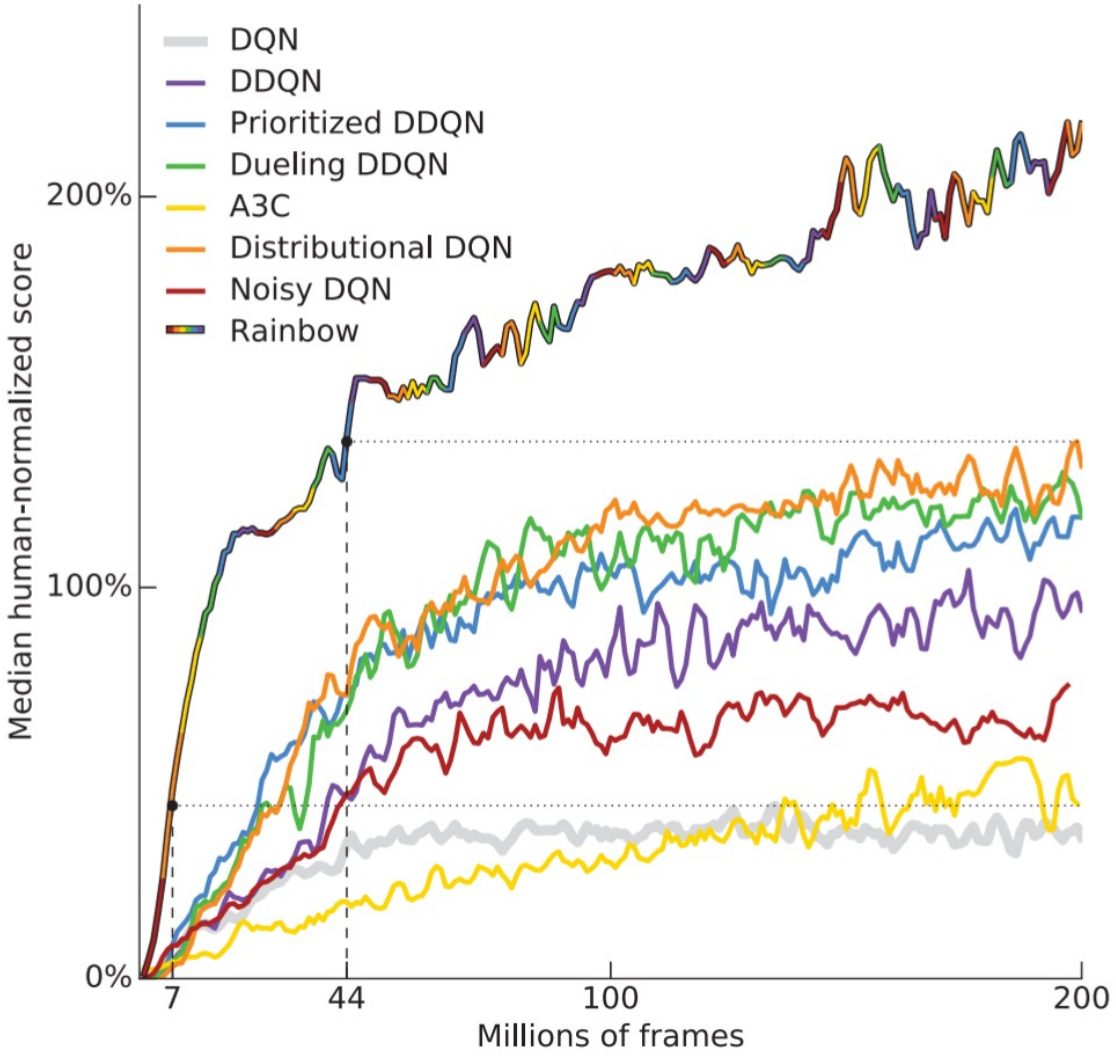
“Double Reinforcement Learning with Double Q-Learning”, van Hasselt et al. AAAI 2016.

- Dueling DQN：精细捕捉价值和行动的细微关联、多种advantage函数建模

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha) - \frac{1}{|A|}\sum_{a'} A(s, a'; \theta, \alpha)$$

“Dueling Network Architectures for Deep Reinforcement Learning”, Wang et al. ICML 2016.

Rainbow: 结合众多Value-based DRL方法



Hessel et al. Rainbow: Combining Improvements in Deep Reinforcement Learning. AAAI 2018.

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
- 基于确定性策略的方法
  - 确定性策略梯度（DPG），DDPG，TD3
- 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

确定性策略梯度

讲师：张伟楠 – 上海交通大学

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

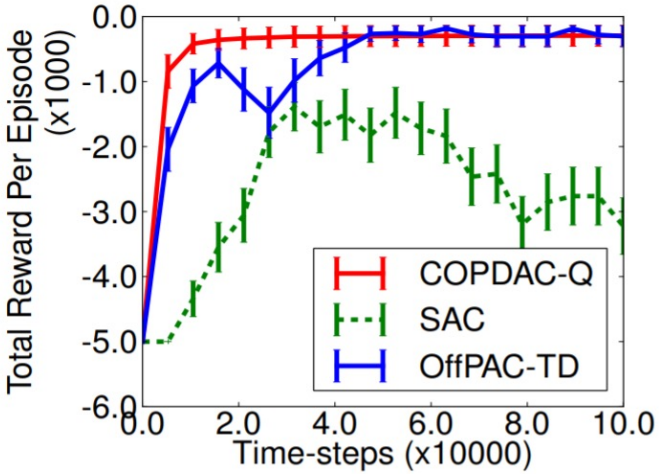
$$J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[Q^w(s,a)]$$

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

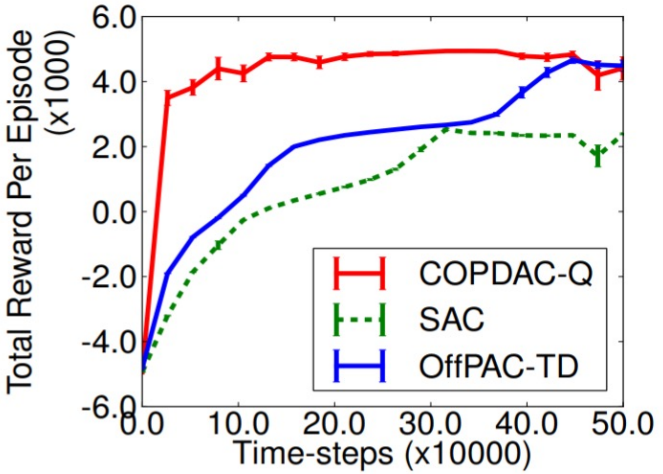
在线策略　　链式法则

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

确定性策略梯度实验效果

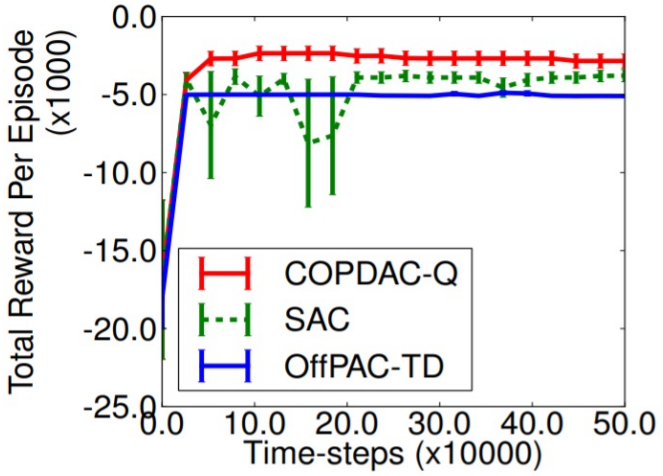


(a) Mountain Car



(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

讲师：张伟楠 – 上海交通大学

DDPG：深度确定性策略梯度

- 对于确定性策略的梯度

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

- 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. NIPS 2015.

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**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ 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 (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled gradient:

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

    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

动作上的噪声

离线策略

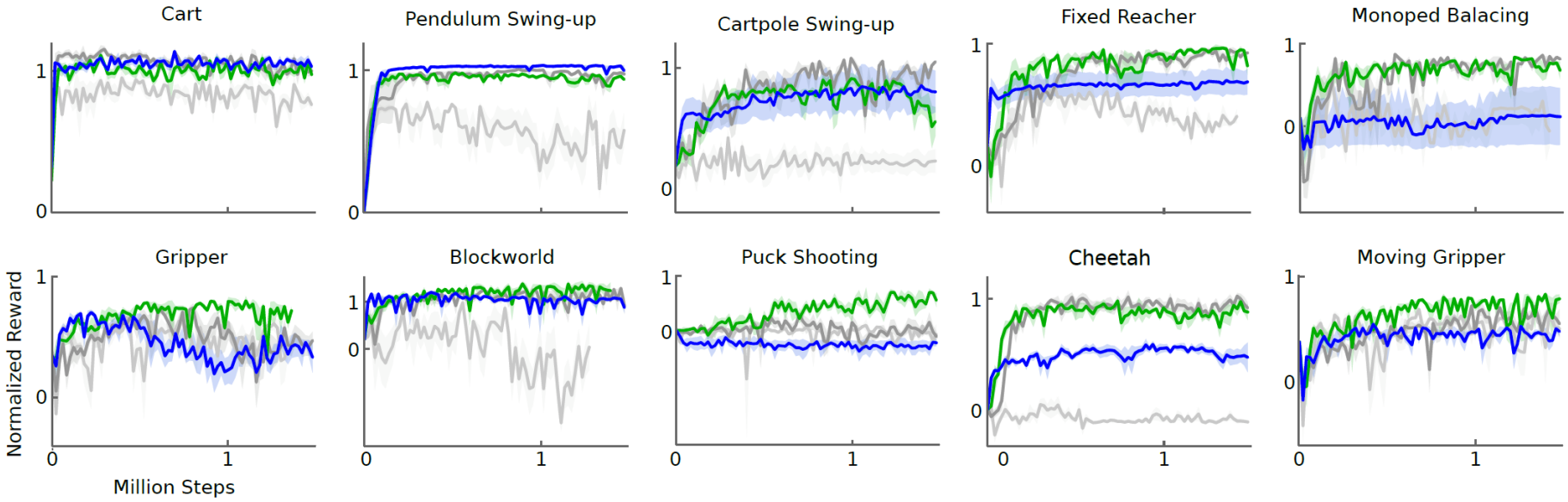
更新critic网络（a_i带有噪声）

目标critic网络

目标actor网络

更新actor网络

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
- 可以看到：目标网络至关重要

Lillicrap et al. Continuous control with deep reinforcement learning. NIPS 2015.

Twin Delayed DDPG (TD3)

双价值函数策略延时更新

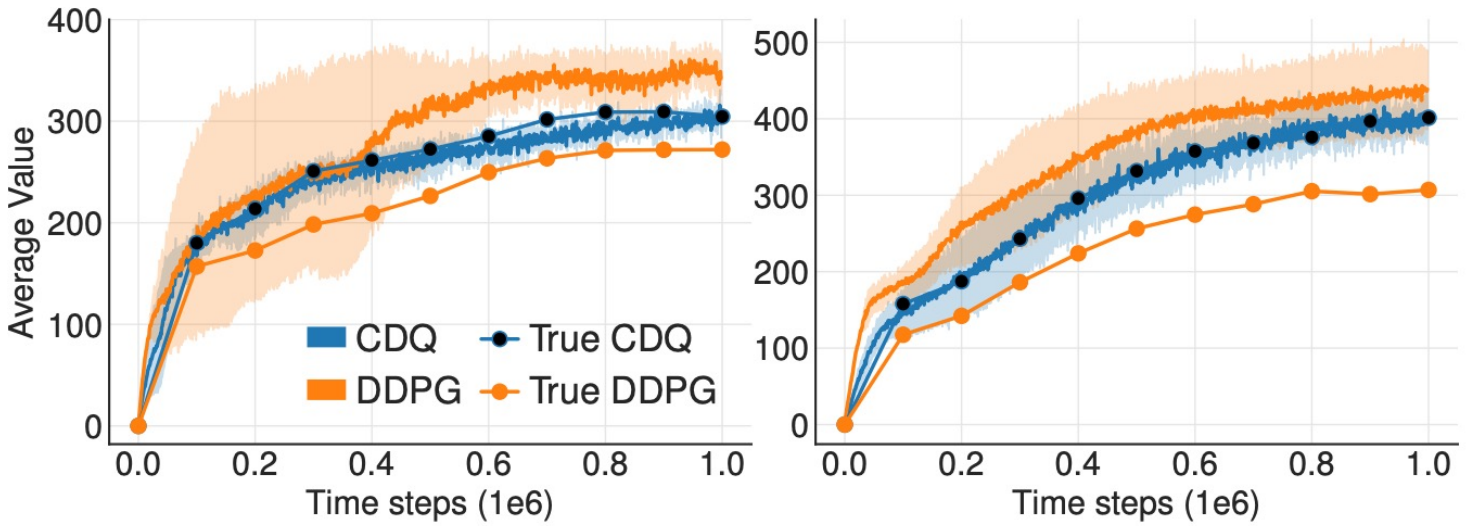
讲师：张伟楠 – 上海交通大学

TD3：双价值函数策略延时更新

- 对于确定性策略的梯度

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

- 在DDPG实际应用中，仍然有对Q函数过高估计问题
  - π可能利用 Q 函数的漏洞（exploitation）



(a) Hopper-v1　　(b) Walker2d-v1

Fujimoto et al. Addressing Function Approximation Error in Actor-Critic Methods. ICML 2018.

TD3：双价值函数策略延时更新

- Twin Delayed DDPG (TD3)

1. 对策略的输出做平滑操作，让策略更难利用到Q函数的漏洞

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

2. 学习两个Q，并使用较小值来做学习目标

$$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s'))$$

$$L(\phi_1, \mathcal{D}) = \mathop{\text{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_1}(s,a) - y(r,s',d)\right)^2\right]$$

$$L(\phi_2, \mathcal{D}) = \mathop{\text{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_2}(s,a) - y(r,s',d)\right)^2\right]$$

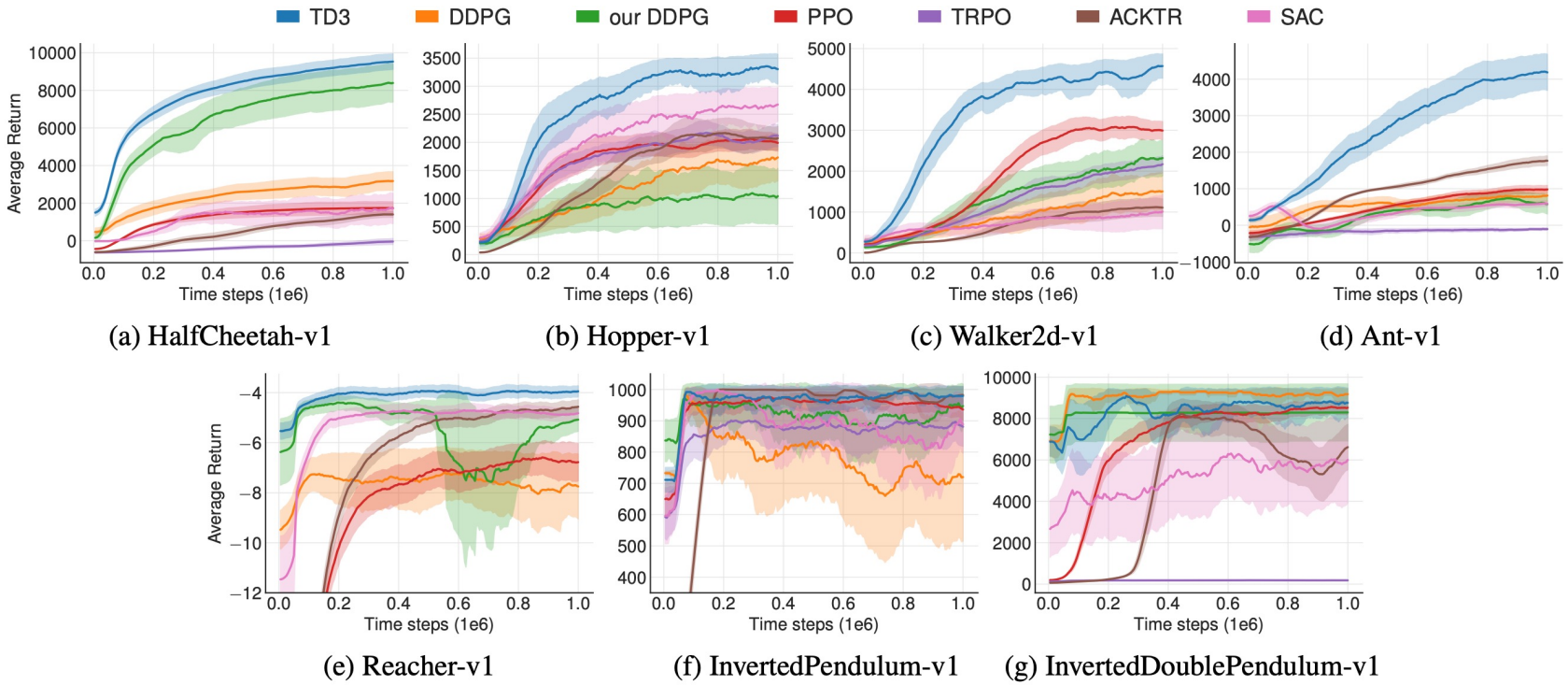
3. 策略学习最大化s处得到的Q价值

$$\max_{\theta} \mathop{\text{E}}_{s\sim\mathcal{D}}\left[Q_{\phi_1}(s, \mu_\theta(s))\right]$$

4. 策略参数更新延迟
   - 一般比Q函数慢1倍，也即是Q函数更新2次，策略更新1次

Fujimoto et al. Addressing Function Approximation Error in Actor-Critic Methods. ICML 2018.

TD3：双价值函数策略延时更新



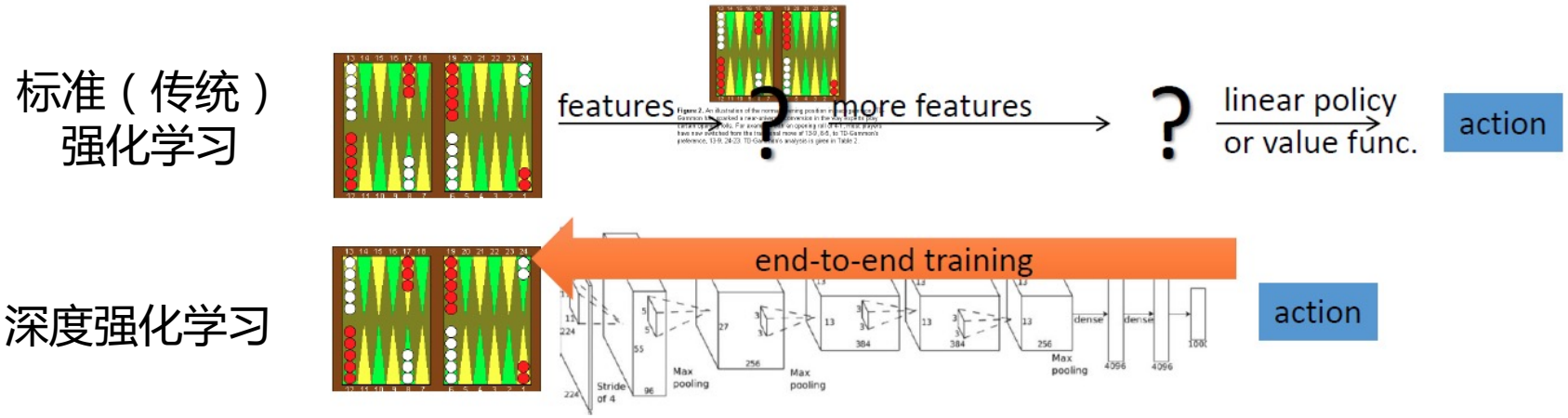
(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

(e) Reacher-v1 (f) InvertedPendulum-v1 (g) InvertedDoublePendulum-v1

Environment	TD3	DDPG	Our DDPG	PPO	TRPO	ACKTR	SAC
HalfCheetah	**9636.95 ± 859.065**	3305.60	8577.29	1795.43	-15.57	1450.46	2347.19
Hopper	**3564.07 ± 114.74**	2020.46	1860.02	2164.70	2471.30	2428.39	2996.66
Walker2d	**4682.82 ± 539.64**	1843.85	3098.11	3317.69	2321.47	1216.70	1283.67
Ant	**4372.44 ± 1000.33**	1005.30	888.77	1083.20	-75.85	1821.94	655.35
Reacher	**-3.60 ± 0.56**	-6.51	**-4.01**	-6.18	-111.43	-4.26	-4.44
InvPendulum	**1000.00 ± 0.00**	**1000.00**	**1000.00**	**1000.00**	985.40	**1000.00**	**1000.00**
InvDoublePendulum	**9337.47 ± 14.96**	**9355.52**	8369.95	8977.94	205.85	9081.92	8487.15

Fujimoto et al. Addressing Function Approximation Error in Actor-Critic Methods. ICML 2018.

深度强化学习总结



- 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
  - DDPG：对价值函数求动作的导数，再根据链式法则对策略参数求倒数
  - TD3：用两个价值函数做clip来缓和过高估计，对策略输出做平滑，策略更新延缓

THANK YOU