

强化学习

第7节

涉及知识点：
深度强化学习、深度Q网络



深度强化学习

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 $V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$
- ▣ 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

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

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

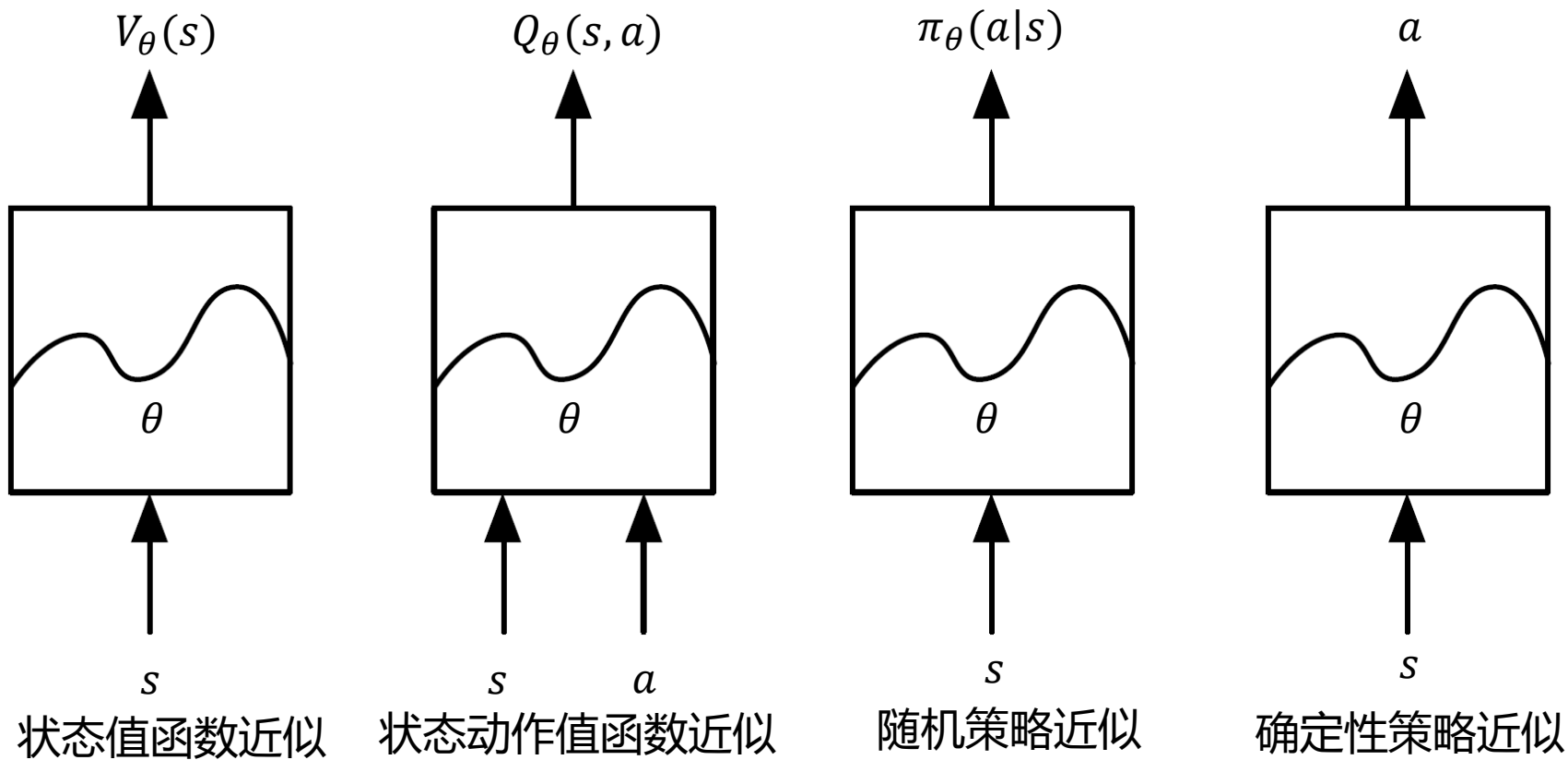
$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta} \left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} \underbrace{Q^{\pi_\theta}(s, a)}_{\text{无梯度}} \right]$$

- 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



深度强化学习

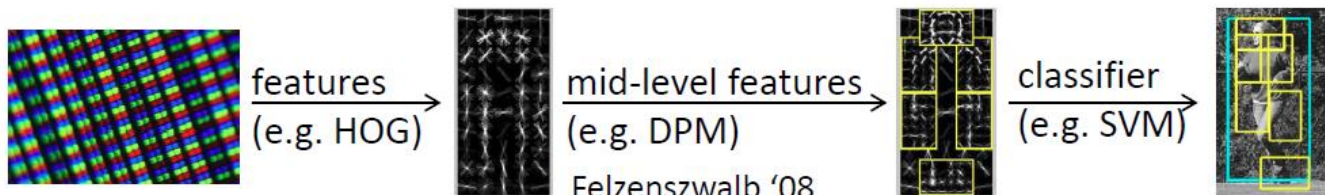
价值和策略近似



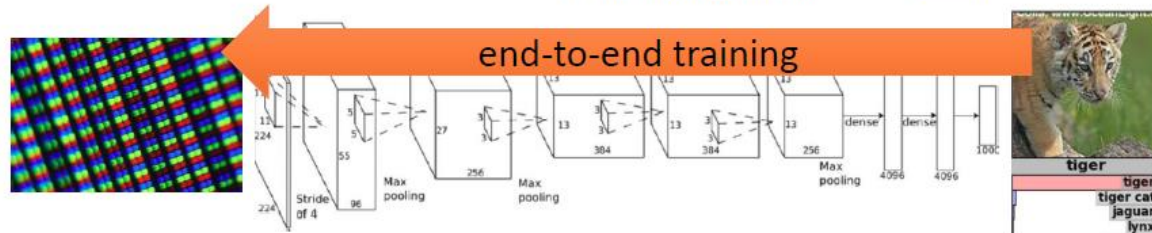
□ 假如我们直接使用深度神经网络建立这些近似函数呢？

端到端强化学习

标准 (传统)
计算机视觉



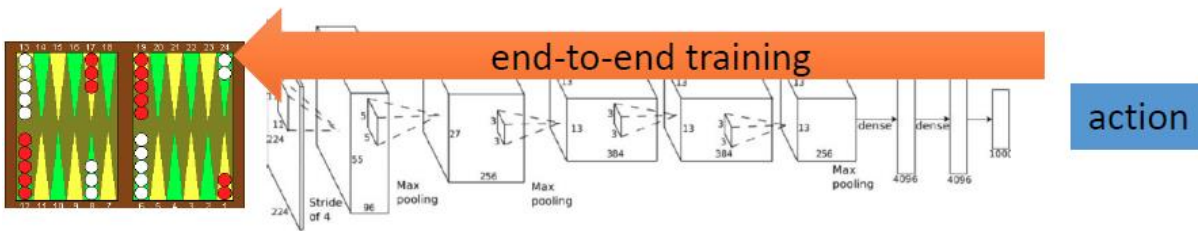
深度学习



标准 (传统)
强化学习



深度强化学习

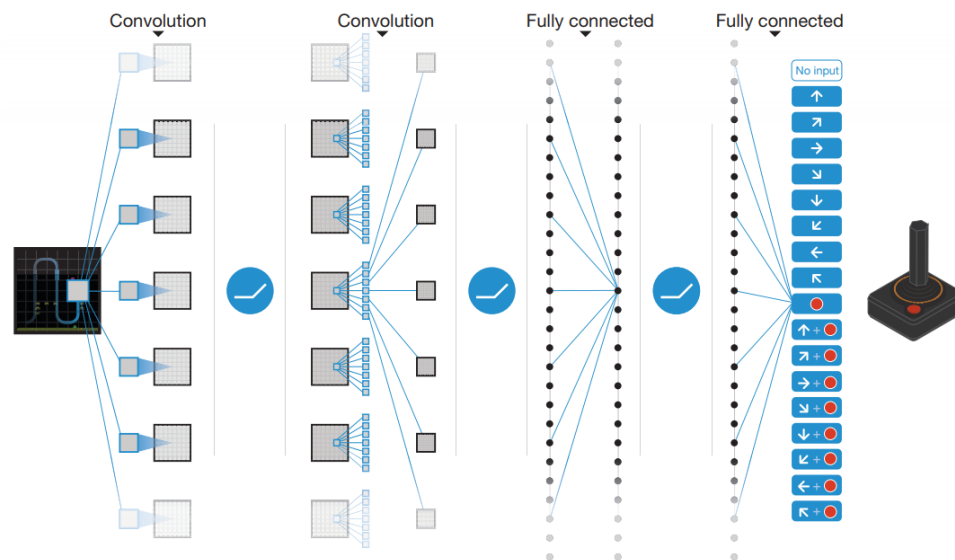


深度强化学习使强化学习算法能够以端到端的方式解决复杂问题

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



深度Q网络

目录

Contents

01 深度Q网络 (DQN)

02 Double DQN

03 Dueling DQN

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

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)

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 $P(t) = \frac{p_t^\alpha}{\sum_k p_k^\alpha}$ 。

□ 重要性采样 (Importance Sampling)

- 权重为 $\omega_t = \frac{(N \times P(t))^{-\beta}}{\max_i \omega_i}$

经验回放

Algorithm 1 Double DQN with proportional prioritization

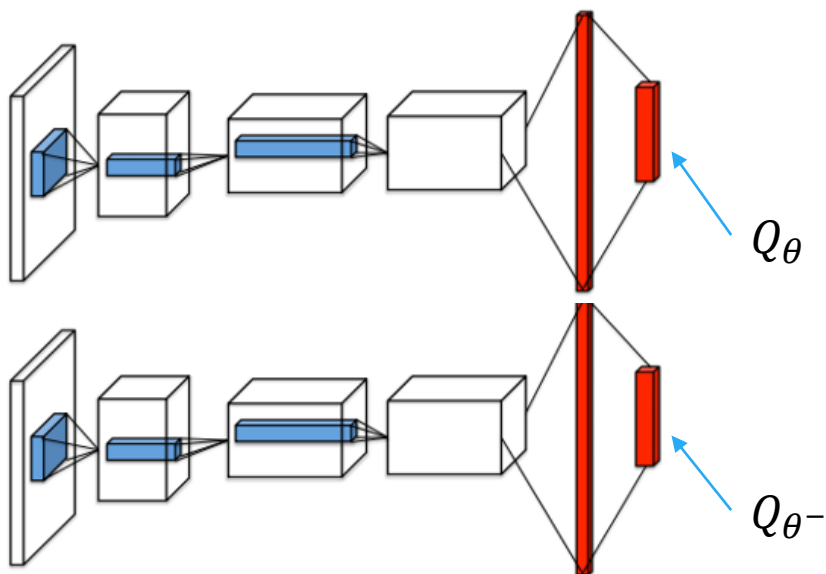
- 1: **Input:** minibatch k , step-size η , replay period K and size N , exponents α and β , budget T .
 - 2: Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
 - 3: Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
 - 4: **for** $t = 1$ **to** T **do**
 - 5: Observe S_t, R_t, γ_t
 - 6: Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in \mathcal{H} with maximal priority $p_t = \max_{i < t} p_i$
 - 7: **if** $t \equiv 0 \pmod K$ **then**
 - 8: **for** $j = 1$ **to** k **do**
 - 9: Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
 - 10: Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
 - 11: Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg \max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
 - 12: Update transition priority $p_j \leftarrow |\delta_j|$ 优先级采样
 - 13: Accumulate weight-change $\Delta \leftarrow \Delta + \underline{w_j} \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$ Importance sampling
优化目标为uniform distri.
 - 14: **end for**
 - 15: Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
 - 16: From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
 - 17: **end if**
 - 18: Choose action $A_t \sim \pi_\theta(S_t)$
 - 19: **end for**
-

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

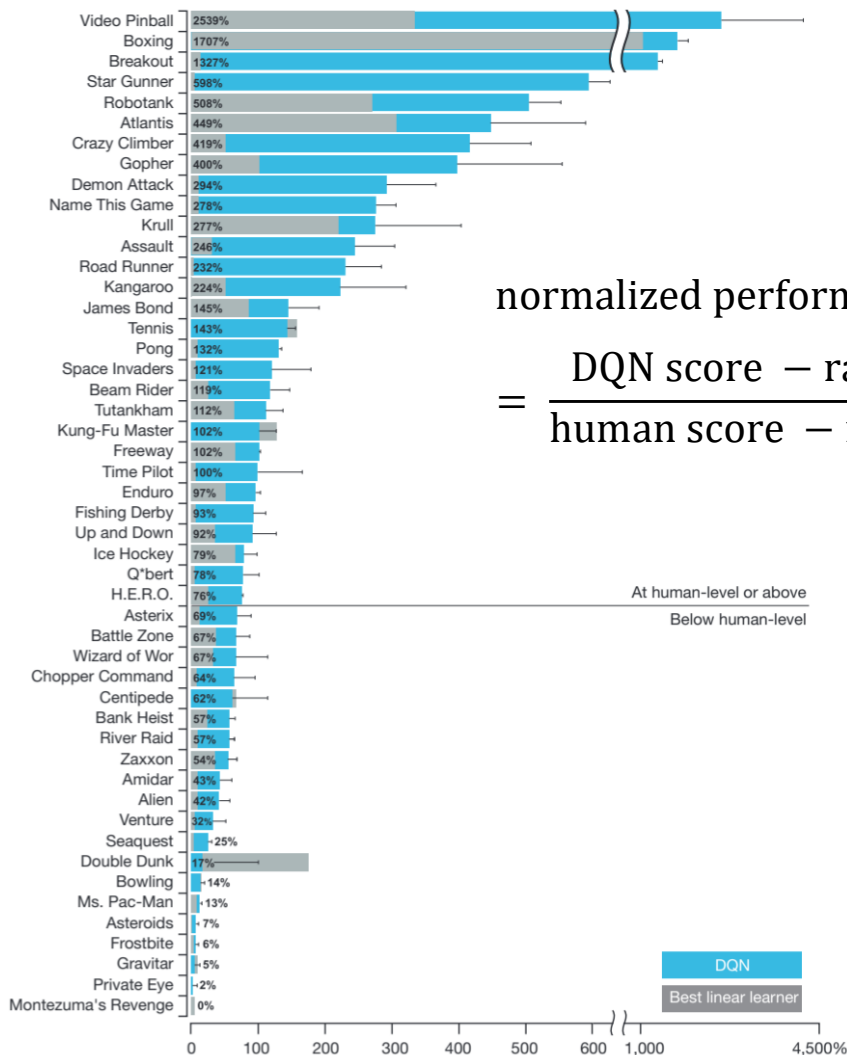
$$L_i(\theta_i) = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t, p_t \sim D} \left[\frac{1}{2} \omega_t \underbrace{\left(r_t + \gamma \max_{a'} Q_{\theta_i^-}(s_{t+1}, a') \right)}_{\text{target}} - Q_{\theta_i}(s_t, a_t) \right]^2$$



算法流程

- 
1. 收集数据：使用 ϵ -greedy 策略进行探索利用，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^- 。

在 Atari 环境中的实验结果

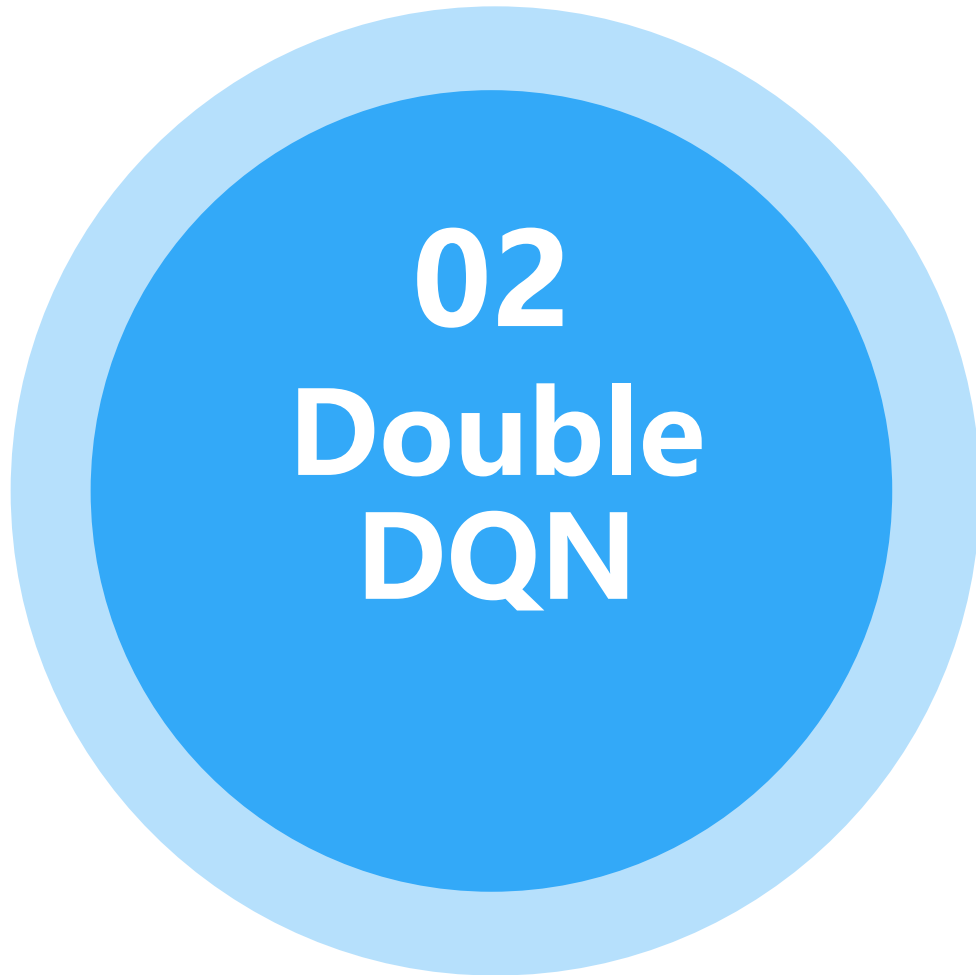


normalized performance

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

At human-level or above
Below human-level

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



02
Double
DQN

Q-learning中的过估计

□ Q函数的过高估计

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

\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

- 假设有随机变量 X_1, X_2 , 有 $\mathbb{E}[\max(X_1, X_2)] \geq \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$

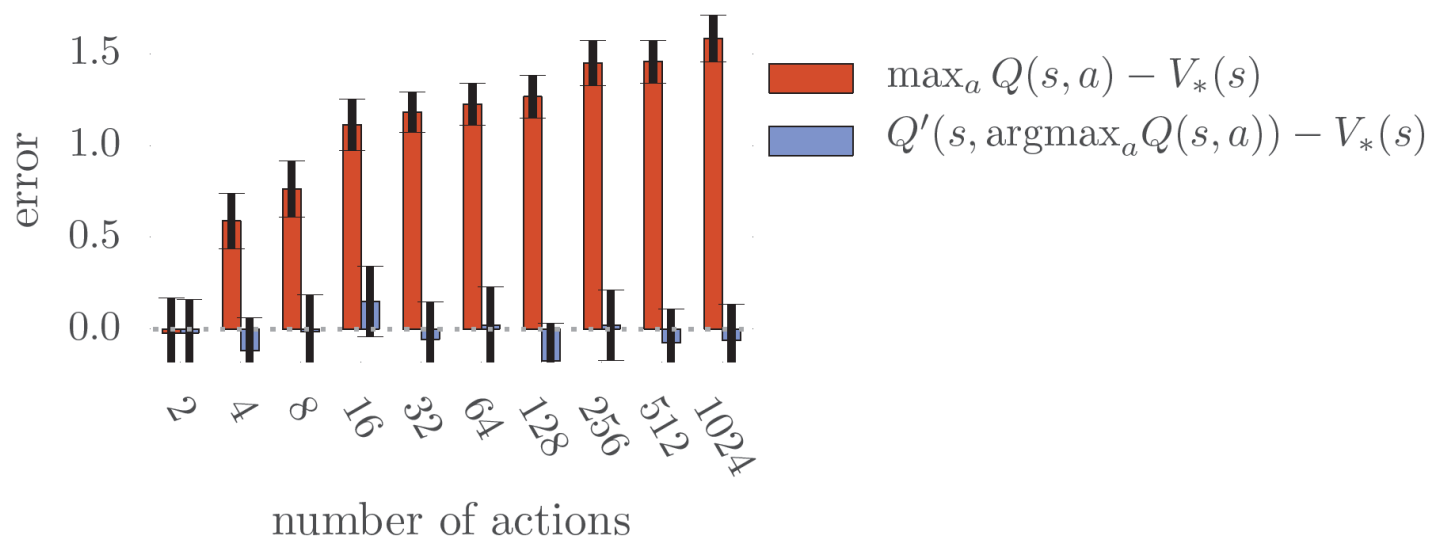
$$\begin{aligned} \max_{a' \in A} Q_{\theta'}(s_{t+1}, a') &= Q_{\theta'}(s_{t+1}, \arg \max_{a'} Q_{\theta'}(s_{t+1}, a')) \\ &= \mathbb{E}[R | s_{t+1}, \arg \max_{a'} Q_{\theta'}(s_{t+1}, a'), \theta'] \\ &\geq \max(\mathbb{E}[R | s_{t+1}, a_1, \theta'], \mathbb{E}[R | s_{t+1}, a_2, \theta'], \dots), a_i \in A \end{aligned}$$

Q 函数的值被视作在状态 s' , 动作 a' 下的回报期望值

我们真正想要得到的最大价值

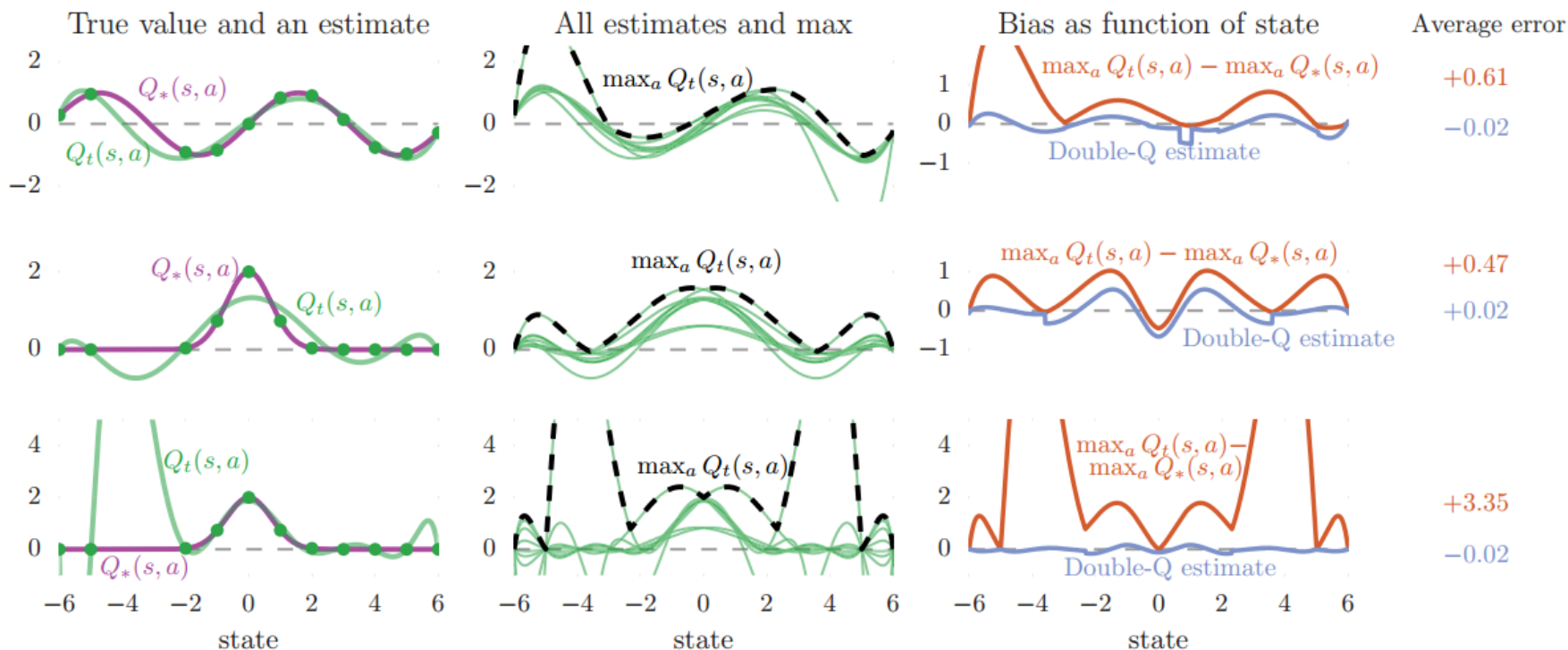
Q-learning中的过估计

- 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

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 DQN

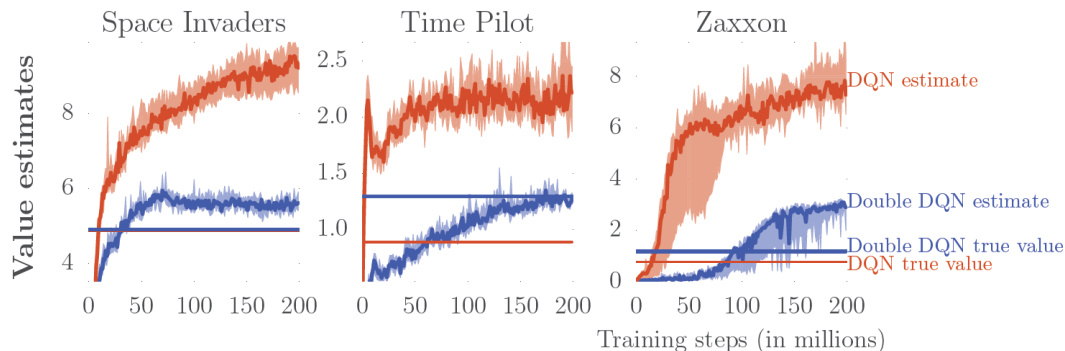
- 使用不同的网络来估值和决策

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

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

在 Atari 环境中的实验结果

价值估计误差

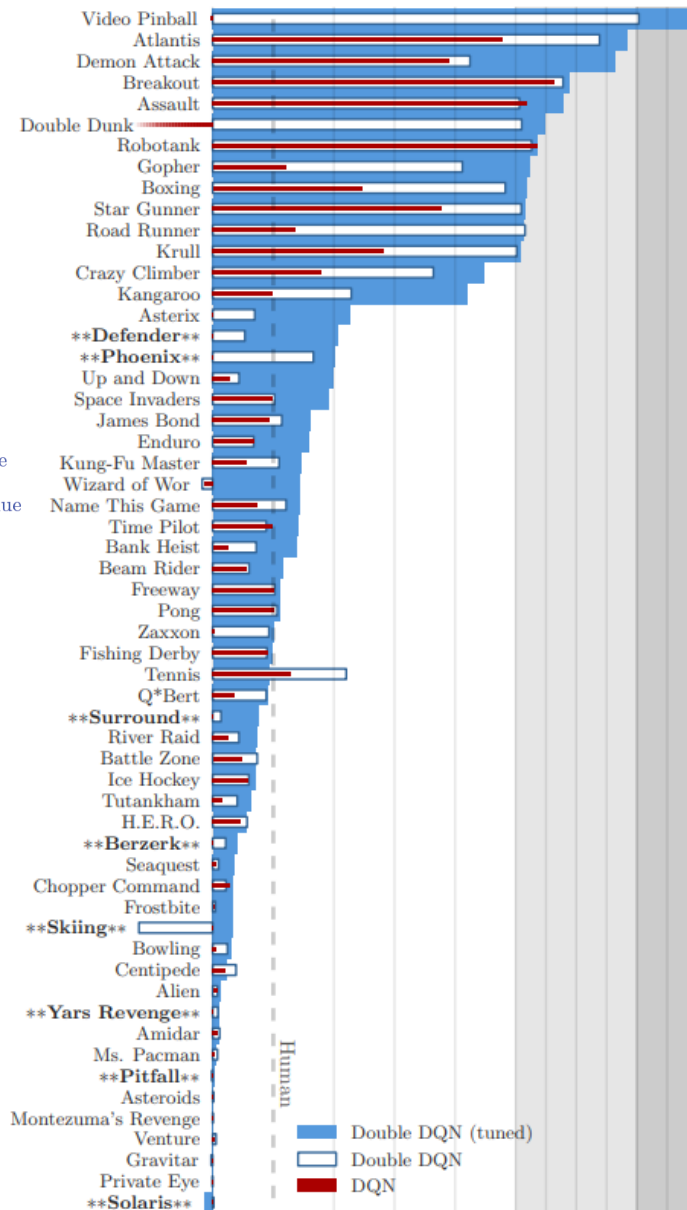


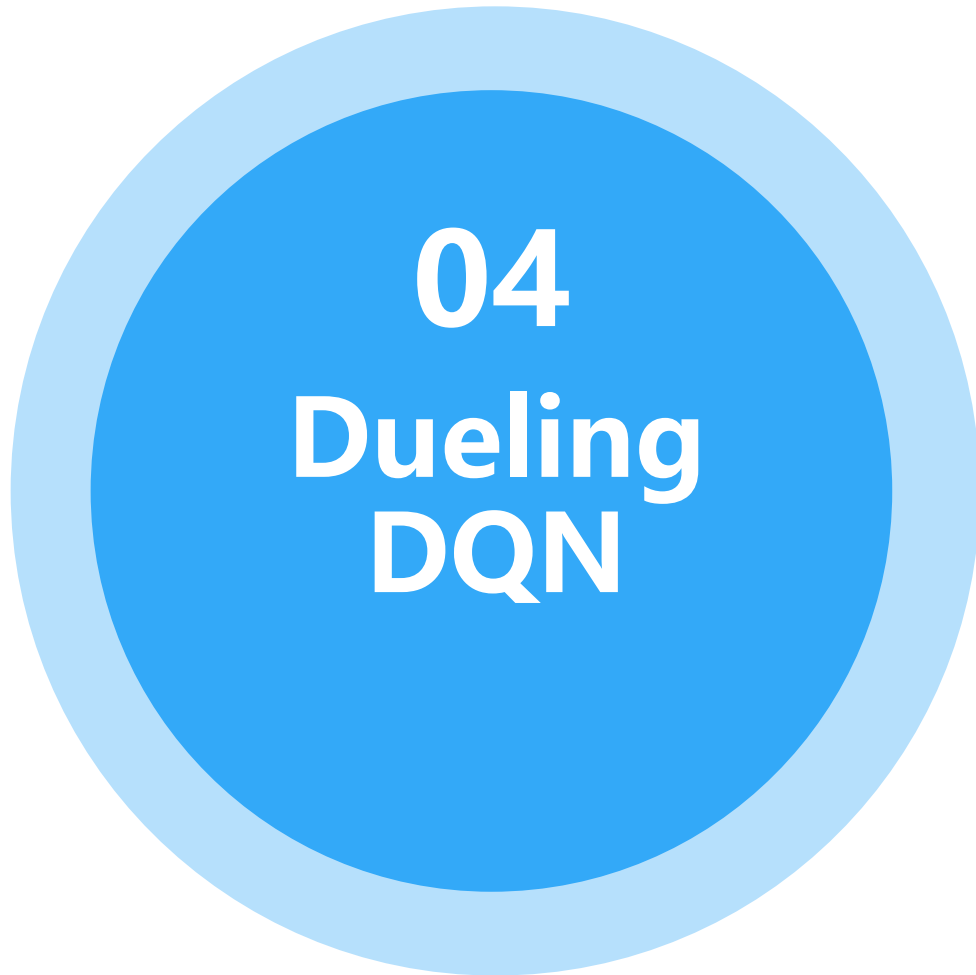
Atari Performance

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

normalized performance

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





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 $\varepsilon(s, a)$? $\rightarrow \varepsilon(s, a) = Q(s, a) - V(s) \rightarrow$ 也称为Advantage函数

Dueling DQN

Advantage 函数

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

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

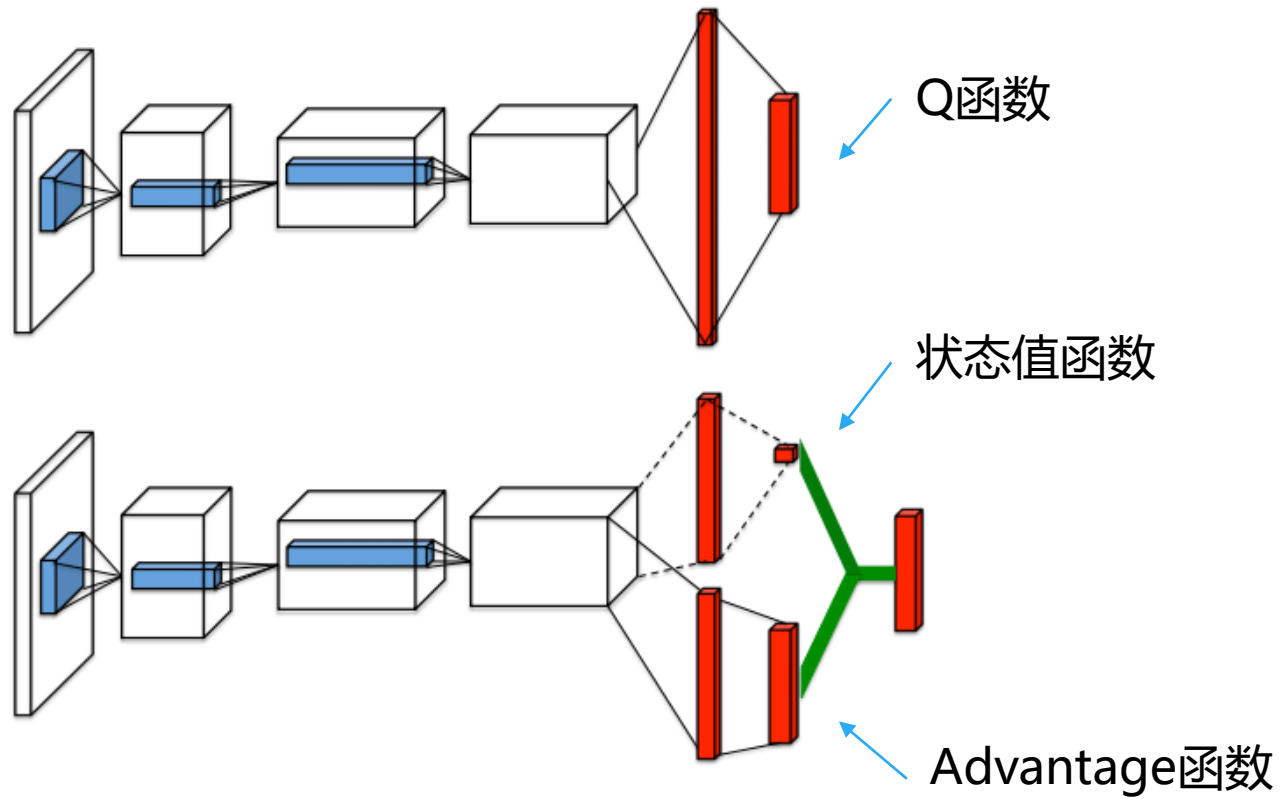
$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)}[Q^\pi(s, a)]$$

不同的Advantage聚合形式

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

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

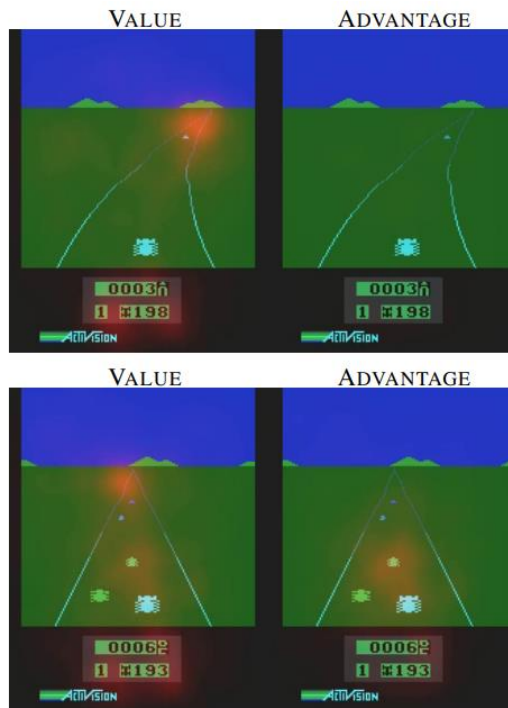
网络结构



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

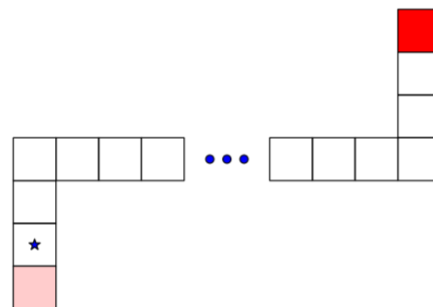
探索任务：走廊环境

□ 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

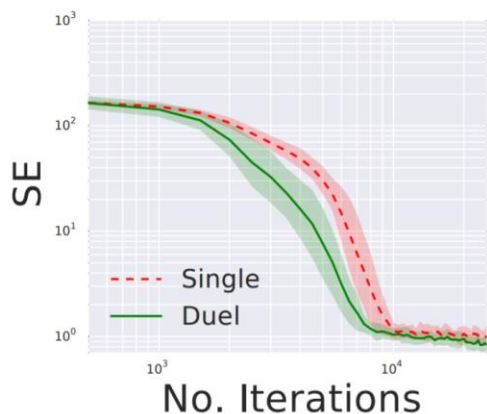
□ Q函数评估均方误差

CORRIDOR ENVIRONMENT

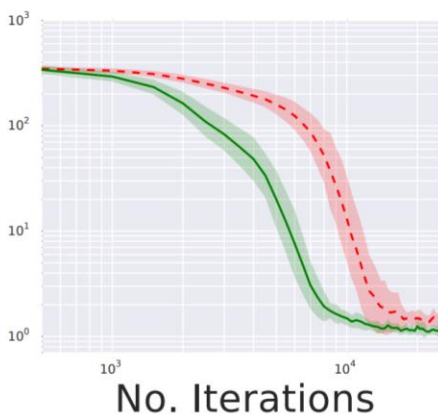


更多的行动是‘不动’

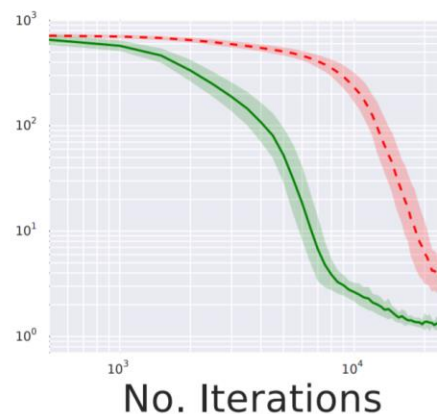
5 ACTIONS



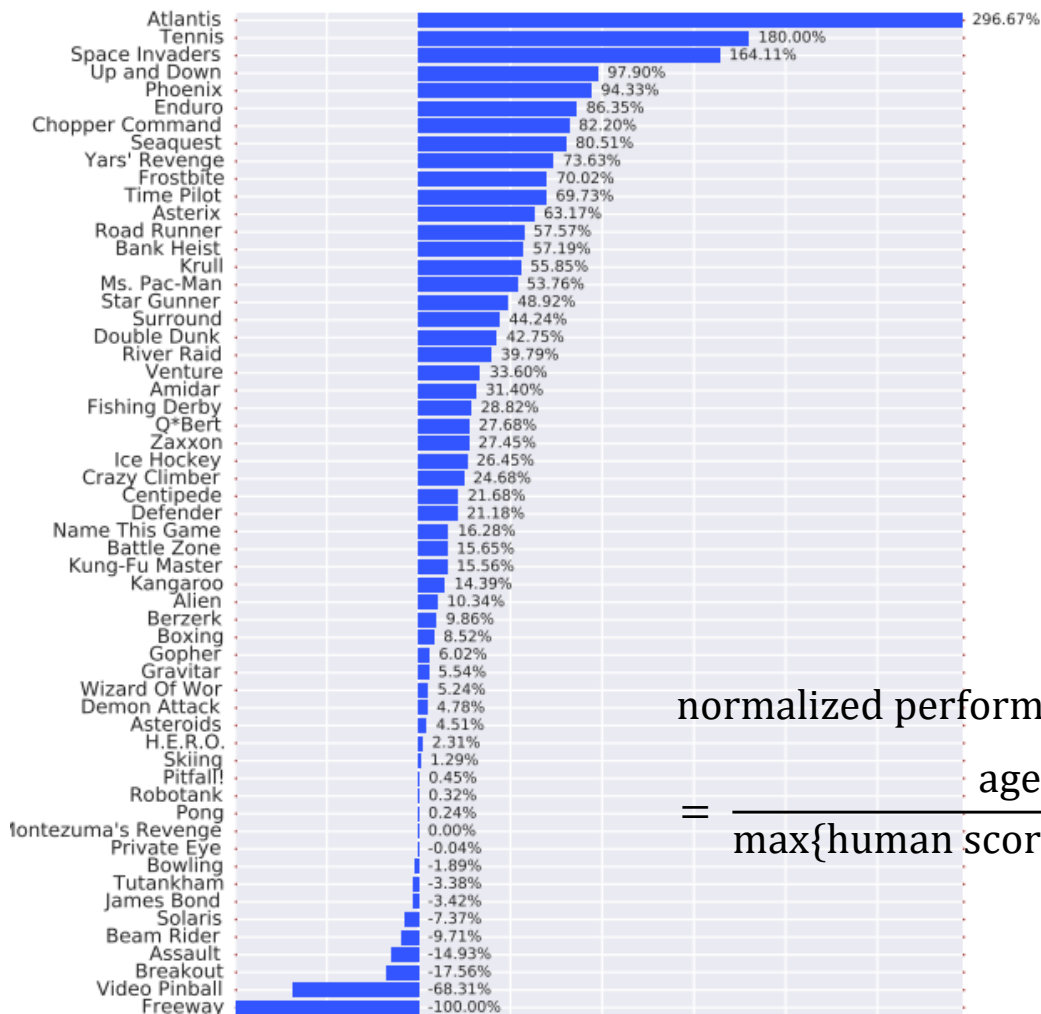
10 ACTIONS



20 ACTIONS



在 Atari 环境中的实验结果I

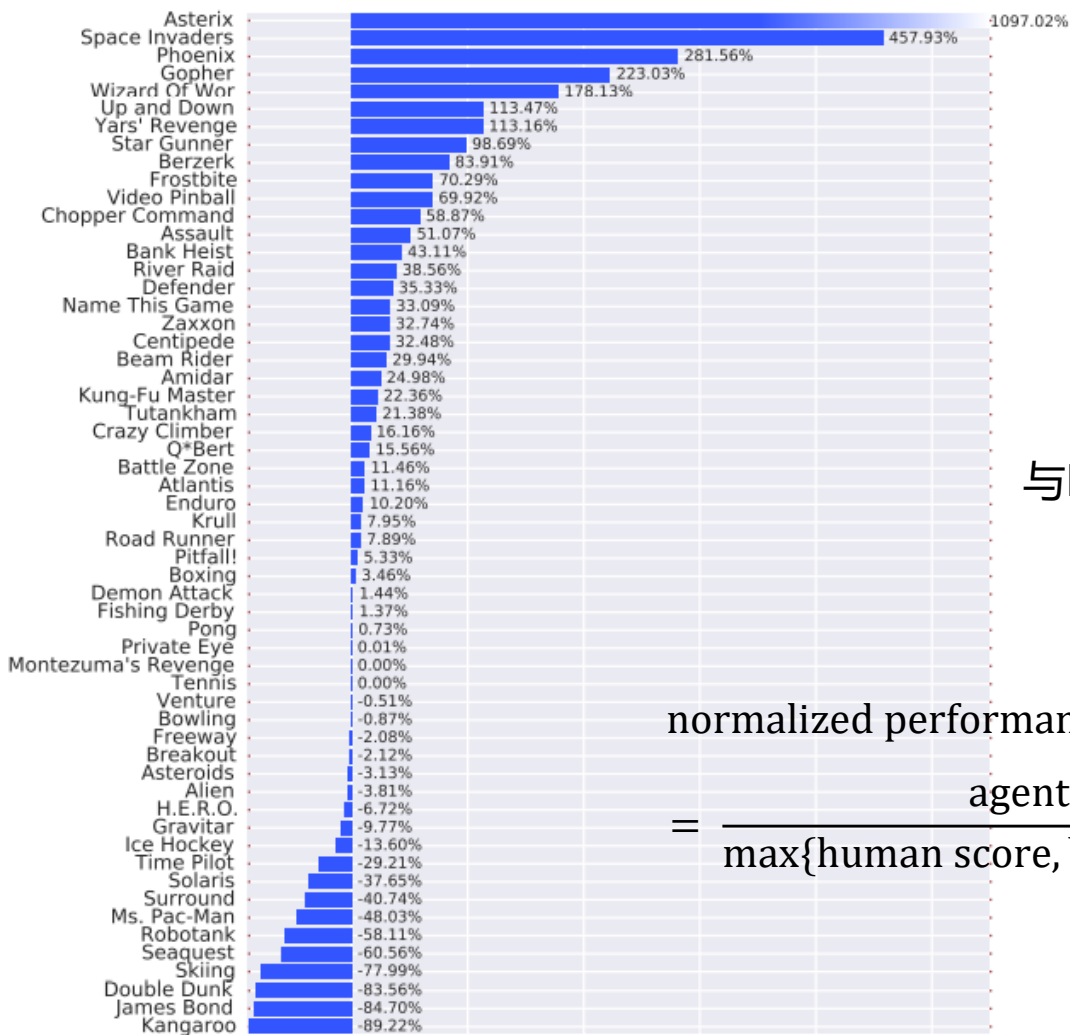


与DQN对比

normalized performance

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

在 Atari 环境中的实验结果II



与Prioritized Double DQN对比

normalized performance

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

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

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

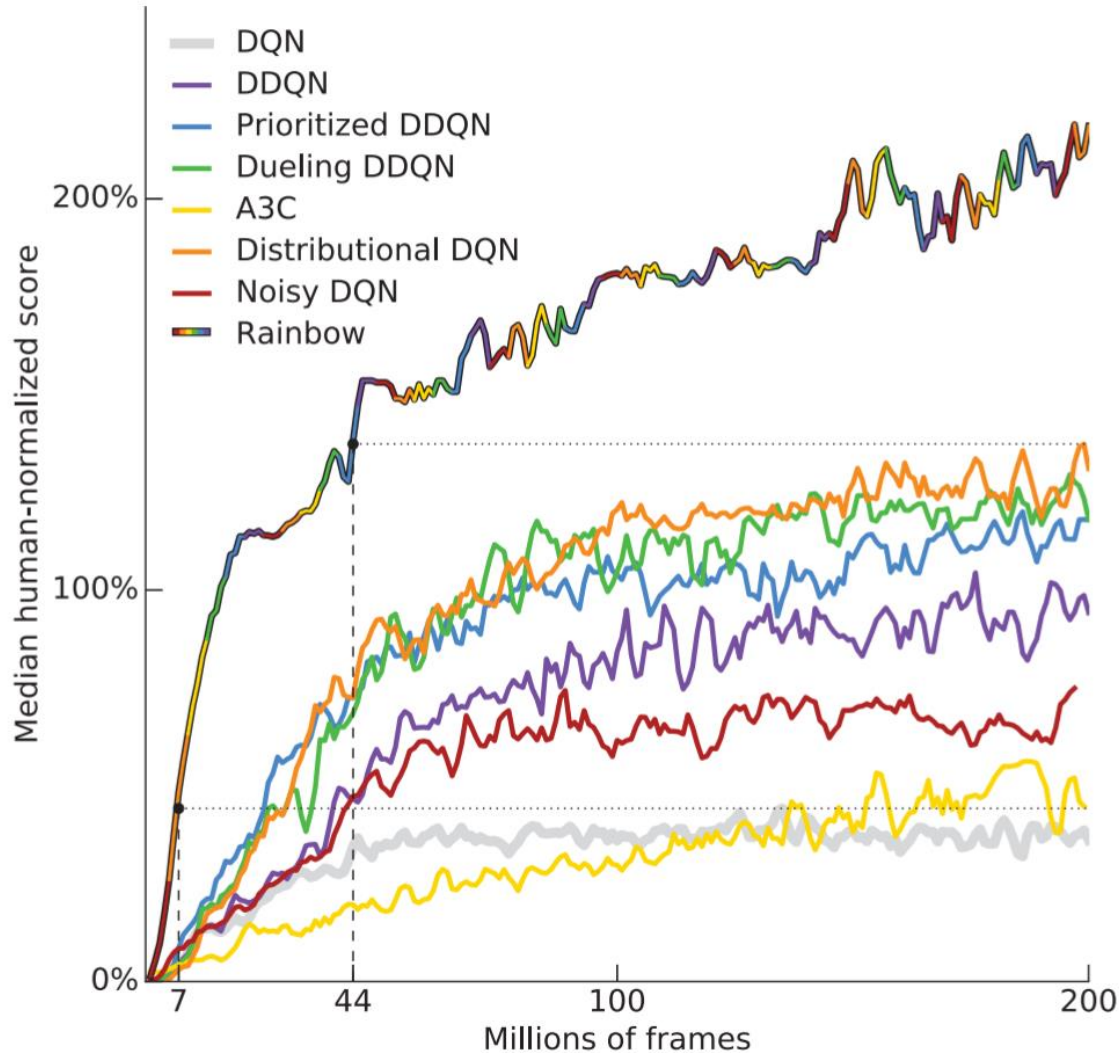
“Double Reinforcement Learning with Double Q-Learning”, van Hasselt et al. AAAI 2016.

- 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



THANK YOU