

JOYRL论文阅读<Rainbow: Combining Improvements in Deep Reinforcement Learning>

作者: Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D.

实验室: Google DeepMind

论文地址: <https://arxiv.org/pdf/1710.02298.pdf>

发表: In *Thirty-second AAAI conference on artificial intelligence.*

1 摘要

1.1 摘要--背景及问题

从DQN¹推导过程中发现依旧存在很多问题, 常见的改进措施Double DQN²、Dueling DQN³、Prioritized replay⁴、Multi-step⁵、Distributional RL⁶、Noisy Net⁷等方法, 这些方法并不是完全独立, 比如在Dueling中其实已经将Double DQN和Prioritized replay结合起来。

1.2 摘要--方法

本文希望将上述六种DQN方法结合经验融合在一起, 来得到一个更好的网络。

1.3 摘要--贡献

1. 成为Atari 2600中SOTA(State-of-the-art)
2. 我们还提供详细的消融研究的结果, 该研究结果显示了每个组件对整体性能的贡献。

2 问题背景

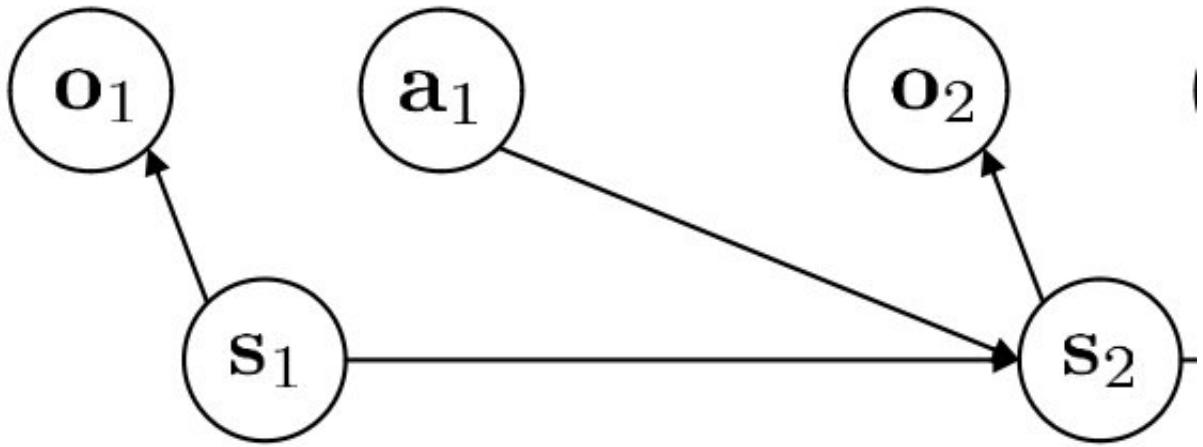
2.1 RL problem & 记号

强化学习希望一个具有动作(Action)的智能体(Agent) 在与环境(Environment)交互的过程可以最大化奖励(Reward), 在这个过程中并不会直接监督式的学习。这里分享另外一种定义:

- 一、Mathematical formalism for learning- based decision making
- 二、Approach for learning decision-making and control from experience

MDP (Markov Decision Process) $\{S, A, T, r, \gamma\}$

在不同的时间步下 $t = 0, 1, 2, \dots$, 环境状态 S_t 提供给智能体一个观测信息 O_t , 通常我们会认为是完全观测(即 $S_t = O_t$), 同时智能体根据观测信息做出动作 A_t , 之后环境给出下一个奖励 R_{t+1} , 奖励的折扣 γ_{t+1} 以及更新状态为 S_{t+1}



在这个过程通常 S, A 是有限的情况, 对于环境来说状态转移 (Stochastic transition function)、奖励方程包括

$$T(s, a, s') = P[S_{t+1=s'} | S_t = s, A_t = a]$$

$$r(s, a) = E[R_{t+1} | S_t = s, A_t = a]$$

对于智能体来说, 根据状态 S_t (或者完全观测下的观测 O_t) 得到动作 A_t 来自于策略 π (Policy), 在序列决策中我们的目标是最大化某个状态采取某个动作的折扣奖励之和

$$P(A_t = a) = \pi_\theta[A_t = a | S_t = s]$$

$$\max G_t = \sum_{k=0}^{\infty} r_t^k R_{t+k+1}$$

我们在利用算法进行梯度提升通常会经过三个步骤:

1. 生成样本
2. 评估模型或者是计算回报
3. 提升策略

2.2 Policy Gradient: 直接提升策略

为了最大化策略的回报, 我们可以直接对 G_t 最大化 (即REINFORCE算法)

REINFORCE algorithm:

- 1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

我们可以利用Baseline、N-steps、Discount、Importance sampling等技巧对算法进行改进。

2.3 Actor-Critic方法：评估汇报(Estimate return)与提升策略分开

也可以引入新的状态价值函数 $V^\pi(s)$ 来结合拟合的方式计算 G_t 之后最大化(A3C)，也可以直接利用 $V^\pi(s)$ 和动作状态价值函数 $Q^\pi(s, a)$ 来进行基于价值函数的学习方法。

$$V^\pi(s) = E_\pi[G_t | S_t = s]$$

$$Q^\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

batch actor-critic algorithm:

- 
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
 2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
 3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

我们可以利用回放池(Replay buffer)、神经网络来学习降低策略梯度中的方差。

2.4 Value-based method 抛弃策略

Policy iteration->Value iteration->Q learning

首先从策略迭代与价值迭代说起，[参考链接](#)可以看作是利用动态规划的方式反应强化学习的过程，两者的区别在于反应的是贝尔曼期望还是贝尔曼最优。

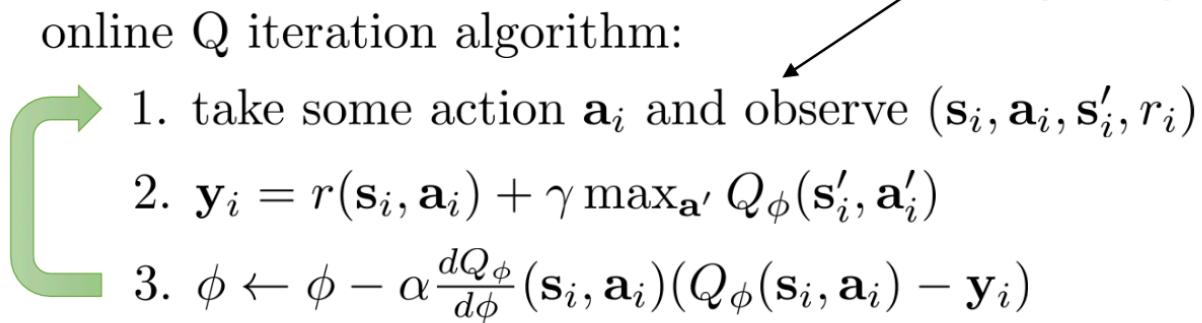
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) \left(r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^\pi(s') \right)$$

$$V^*(s) = \max_{a \in \mathcal{A}} \{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \}$$

在基于价值学习算法的过程中，优点是我们只需要一个经验回放池，只需要 (s, a, s', r) 而不是需要完整的决策序列。我们通常会引入随机探索 (Exploration) 的概念，常见的包括 $\epsilon - Greedy$ 的方法，在一定概率下选择非策略产生的动作。

$$a * t = \begin{cases} \arg \max * a \in \mathcal{A} \hat{Q}(a), & \text{采样概率: } 1-\epsilon \\ \text{从 } \mathcal{A} \text{ 中随机选择,} & \text{采样概率: } \epsilon \end{cases}$$

在价值迭代的基础上，我们可以抛弃对 $V(s)$ 的学习，而只是记录 $Q(s, a)$; Q-iteration algorithm (或者 $Q - learning$) 的过程如下图所示



2.5 DQN推导

在上述我们认识对于MDP目标，从显式表达策略，到结合状态价值再到之后的完全使用价值函数来使用学习的方法，我们仍然没有解决的问题包括

1. 状态-动作空间的连续性
2. 在状态空间和动作空间纬度大的时候无法准确刻画状态-动作价值

随着神经网络的发展，我们希望利用网络拟合的方式来解决上述问题，同时每一步利用 $\epsilon - Greedy$ 的方式来探索回放池中的经验，并利用梯度下降等方法最小化价值函数表达的回报

$$\min(R_{t+1} + \gamma_{t+1} \max_{a'}(S_{t+1}, a') - q_\theta(S_t, A_t))^2$$

- 1 初始化大小为 N 的经验回放池 (PS: 注意有大小限制)
- 2 用相同随机的网络参数初始化 $Q_\theta(s, a)$ 与目标网络 $Q_{\theta'}(s, a)$
- 3 FOR 回合 = 1, N DO:
 - 4 获取环境初始状态 s_1
 - 5 FOR 时间步 = 1, T DO:
 - 6 根据当前网络 $a_t = \max_a Q_\theta(s, a)$ 结合 $\epsilon - Greedy$ 方法来得到 a_t (PS: 注意这一步动作的确定隐含之后DQN回报偏大的特点)
 - 7 执行动作 a_t , 获取 r_t , 环境状态变为 s_{t+1}
 - 8 存储上述采样信息到经验回放池
 - 9 if 经验回放池数目足够:
 - 10 采样 batchsize 个样本 $\{s_t^i, a_t^i, r_t^i, s_{t+1}^i\}$
 - 11 计算目标值 $y_t^i = r_t^i + \gamma * \max_a Q_{\hat{\theta}}(s_{t+1}^i, a_t^i)$
 - 12 最小化损失函数 $L = \frac{1}{N} (y_t^i - Q_\theta(s_t^i, a_t^i))^2$
 - 13 更新网络参数
 - 14 END FOR
 - 15 更新目标网络参数
 - 16 END FOR

Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
    end for
end for


---

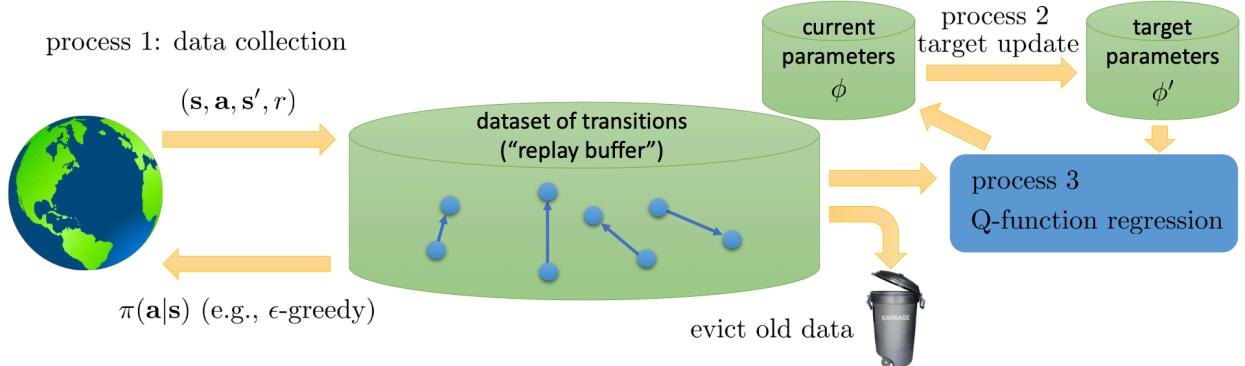

```

其中非常有趣的技巧包括：

1. 经验回放(Experience replay)与随机探索；这一部分主要是为了提高样本采样效率，同时降低后续梯度下降中样本的相关性。
2. 目标网络(Target network)；由于TD误差在策略改变过程中也会改变，因此造成神经网络拟合过程的不稳定性，因此构建新的目标网络，在每次迭代过程中暂时固定，在回合结合后更新参数，这样需要两层Q网络。

[相关代码实现](#)

A more general view



3 DQN改进

虽然DQN成功让强化学习在某些方面超过人类，但是依旧有这许多限制。

3.1 改进1：Double Q- Learning

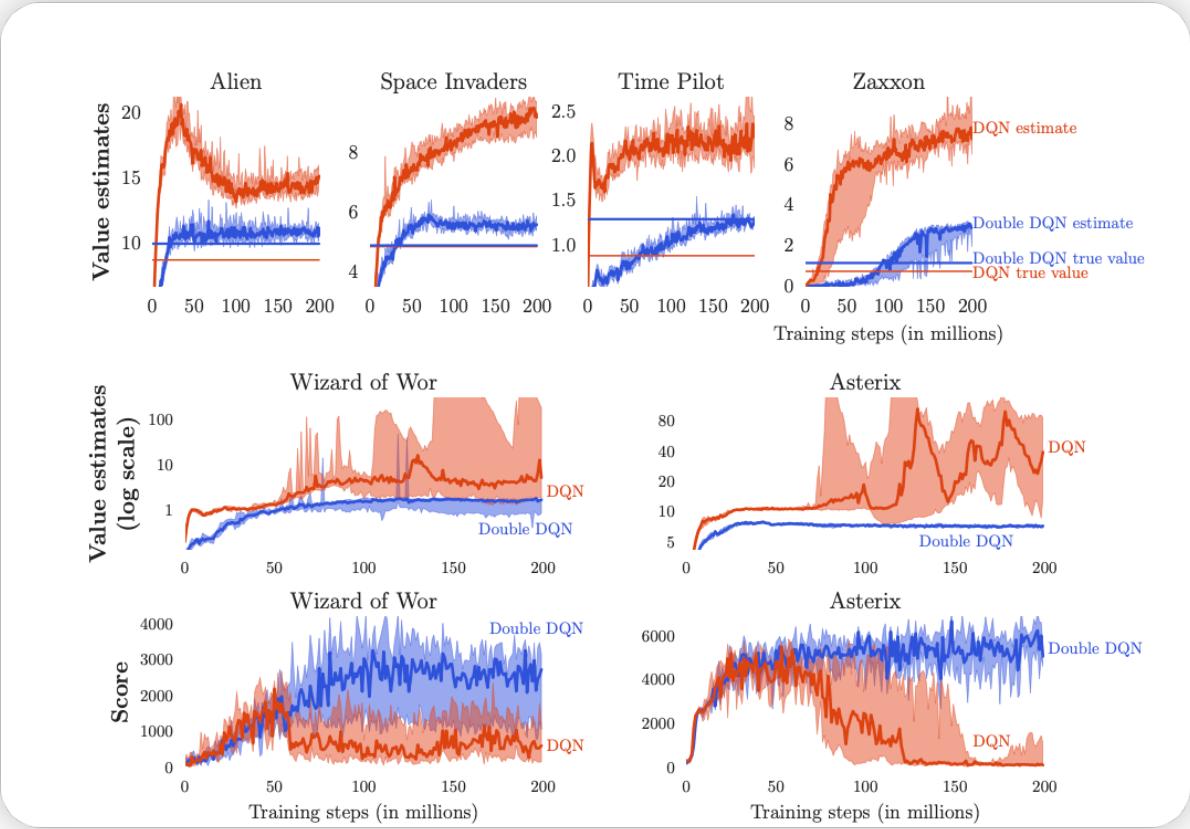
在DQN中会有估计值过高的情况，证明如下：

$$\text{target-value} : y_t^i = r_t^i + \gamma \max_{a_{t+1}^i} Q_{\hat{\theta}}(S_{t+1}^i, a_{t+1}^i)$$

$$\max_{a_{t+1}^i} Q_{\hat{\theta}}(S_{t+1}^i, a_{t+1}^i) = Q_{\hat{\theta}}(s_{t+1}^i, \arg \max_{a_{t+1}^i} Q_{\hat{\theta}}(s_{t+1}^i, a_{t+1}^i))$$

根据期望公式

$$E[\max(X_1, X_2)] > \max(E(X_1), E(X_2))$$



我们通过证明发现估计值较大的原因是因为我在模型选择行为和计算Q值使用同一个网络，如果降低行为选择和Q值计算的相关性就可以降低高估，因此直觉的我们可以设计两个网络

$$Q_{\theta_A}(s, a) = r + \gamma Q_{\theta_b}(s', a')$$

$$Q_{\theta_B}(s, a) = r + \gamma Q_{\theta_a}(s', a')$$

我们的确可以新加一个网络，但是会增加学习难度，需要重新设计架构。所以为什么不直接使用 $Q_{\theta}(s, a)$ 作为行为的估计？

$$(\text{target_value_double_Q - learning}) : y_t^i = r_t^i + \gamma Q_{\hat{\theta}}(s_{t+1}^i, \text{argmax}_{r_{t+1}^i Q_{\theta}}(s_{t+1}^i, a_{t+1}^i))$$

3.2 改进2: Prioritized replay

在DQN学习中为高效利用 (s, a, r, s) 样本，我们会使用经验回放的方式来存储一定规模的样本，在梯度下降的时候通常是从经验回放中均匀采样 (Uniformly sampling) 来进行学习，但是我们依旧会存在两个问题：

1. 依旧没有完全解决数据之间独立同分布的假设
2. 容易忘记一些罕见的、重要的经验数据

在该论文中作者首先制定指标“TD-error”作为衡量 $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ 的信息量大小，作为采样的优先级，同时利用随机优先级采样、偏置和重要性采样等方式来避免贪心的问题。优先级的获取有3.2.1和3.2.2两种方式

$$(\text{随机采样}) P(i) = \frac{p_i^\alpha}{\sum_k P_k^\alpha}$$

3.2.1 比例优先级 (Proportional prioritization)

$$P_i = |\sigma_i| + \epsilon$$

3.2.2 基于排名的优先级(Rank-based prioritization)

$$P_i = \frac{1}{\text{rank}(i)}; \text{ 优点可以保证线性性质, 对异常值不敏感。}$$

上述两种是不同得到重要性的方式；在实现时候采用Sum-tree的数据结构降低算法复杂程度。在采样中考虑重要性采样(Importance sampling)，并由此来进行热偏置(Annealing the bias) 来修正误差

$$w_j = \left(\frac{1}{N} * \frac{1}{P(i)} \right)^\beta$$

Algorithm 1 Double DQN with proportional prioritization

```

1: Input: minibatch  $k$ , step-size  $\eta$ , replay period  $K$  and size  $N$ , exponents  $\alpha$  and  $\beta$ , 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  $S_t, R_t, \gamma_t$ 
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  $p_j \leftarrow |\delta_j|$ 
13:      Accumulate weight-change  $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$ 
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

```

3.3 改进3: Dueling networks

Dueling DQN是一种针对基于价值函数的强化学习的网络结构设计，其并不直接输出状态动作价值函数，而是输出状态价值函数与动作状态优势函数，因为通常会共用前几层的卷积参数，在后面则是状态价值函数与优势函数各自的参数。

$$q_\theta(s, a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s), a) - \frac{\sum_{a'} a_\psi(f_\xi(s), a')}{N_{\text{actions}}},$$

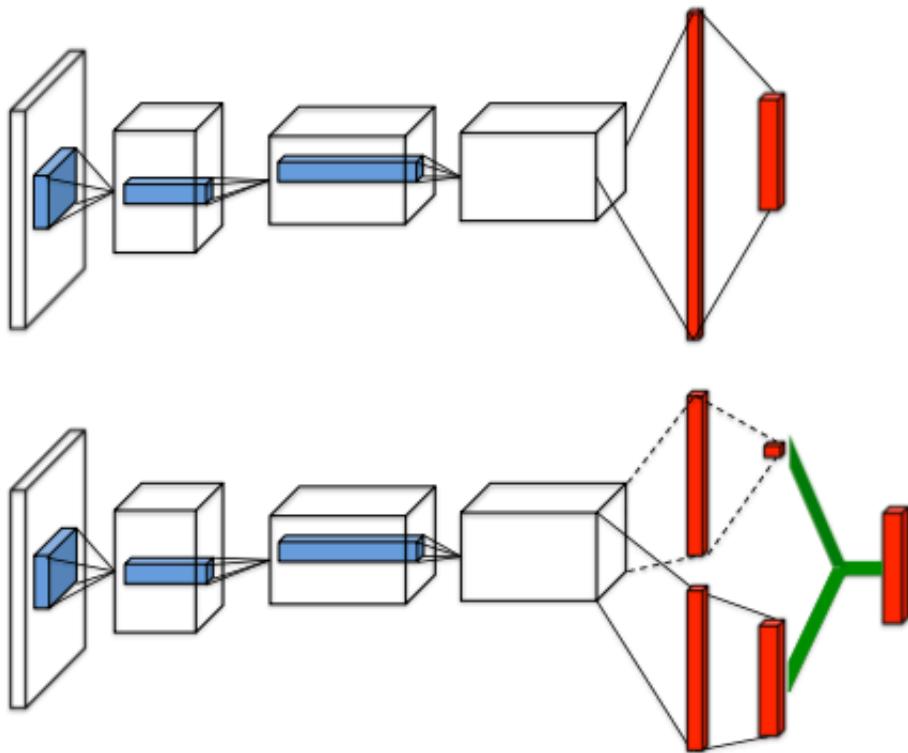


Figure 1. A popular single stream Q -network (**top**) and the dueling Q -network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q -values for each action.

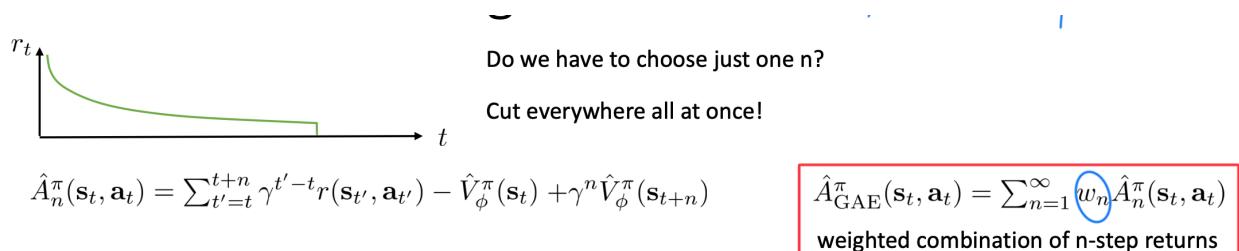
3.4 改进4：Multi-step learning

在对状态动作函数的优势估计时候，通常我们会分为蒙特卡洛方法与Bootstrap(或者是Actor- critic内的C)的方法

$$(MC - sampling) A_t = \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} - b)$$

$$(bootstrap - sampling) G_t = r_t + \gamma * V(s_{t+1}) - V(s_t)$$

前者方法偏差低但是方差较大；后者方差低但是有偏。因此结合两者我们通常会有Multi-step target的方法。



同样的也可以用在DQN中对于状态动作价值函数的估计：

$$R_t^n = \sum_{k=0}^{n-1} \gamma_t^k R_{t+k+1}$$

更新之后的损失函数为

$$(R_t^n + \gamma_t^n \max_{a'} Q_{\hat{\theta}}(S_{t+n}, a') - Q_{\theta}(S_t, A_t))^2$$

3.5 改进5: Distributional RL

在基于价值函数的学习中我们通常是返回一个期望或者最大值而丢失很多其他信息, 因此Distributional RL尝试利用其分布而不是单个值来进行强化学习。首先本文尝试将价值函数范围 $[V_{min}, V_{max}]$ 划分为N个各自来估计价值函数, 利用Boltzmann分布表示价值函数的分布, 同时利用投影的操作

$$(\Phi \hat{\mathcal{T}} Z_{\theta}(x, a))_i = \sum_{j=0}^{N-1} \left[1 - \frac{|[\hat{\mathcal{T}} z_j]_{V_{min}}^{V_{max}} - z_i|}{\Delta z} \right]_0^1 p_j(x', \pi(x')), \quad (7)$$

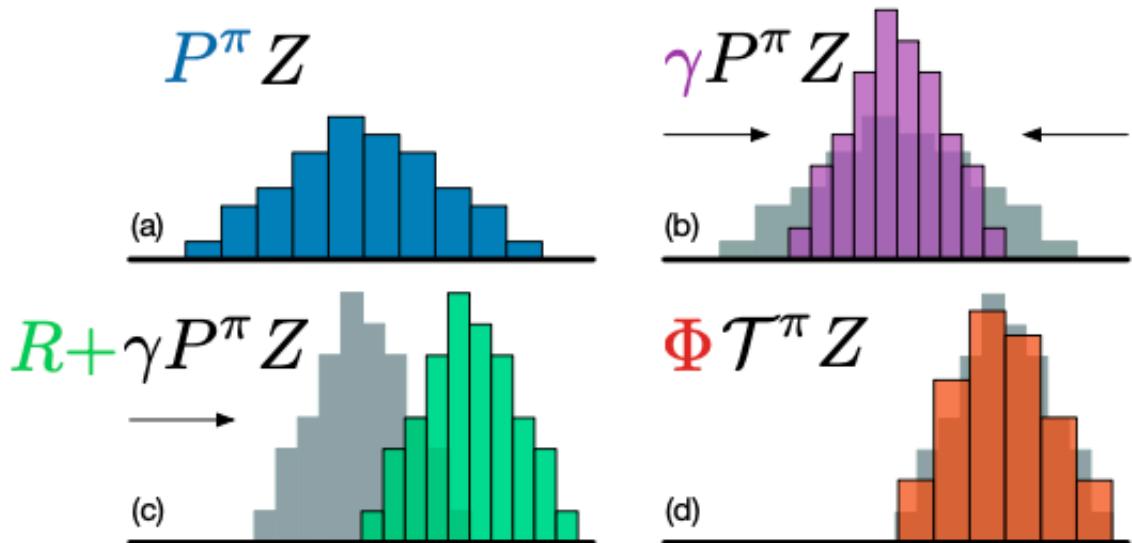


Figure 1. A distributional Bellman operator with a deterministic reward function: (a) Next state distribution under policy π , (b) Discounting shrinks the distribution towards 0, (c) The reward shifts it, and (d) Projection step (Section 4).

由此对于分布拟合可以划分为交叉熵的形式, 算法流程

Algorithm 1 Categorical Algorithm

input A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$

$$Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$$

$$a^* \leftarrow \arg \max_a Q(x_{t+1}, a)$$

$$m_i = 0, \quad i \in 0, \dots, N - 1$$

for $j \in 0, \dots, N - 1$ **do**

Compute the projection of $\hat{T}z_j$ onto the support $\{z_i\}$

$$\hat{T}z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$$

$$b_j \leftarrow (\hat{T}z_j - V_{\text{MIN}})/\Delta z \quad \# b_j \in [0, N - 1]$$

$$l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$$

Distribute probability of $\hat{T}z_j$

$$m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$$

$$m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$$

end for

output $-\sum_i m_i \log p_i(x_t, a_t)$ # Cross-entropy loss

3.6 改进6: Noisy Nets

在Q-learning或者是DQN中，我们的轨迹并不是完全采样的，而是与我们的探索策略相关，最原本的是 ϵ -Greedy策略，这里提出一种NoisyNet来对参数增加噪声来增加模型的探索能力

$$y = (Wx + b) + (b_{\text{noisy}} \odot \epsilon^b + W_{\text{noisy}} \odot \epsilon^w)x$$

噪声的生成可以分为Independent Gaussian noise; Factorised Gaussian noise两种方式。

	Baseline		NoisyNet		Improvement (On median)
	Mean	Median	Mean	Median	
DQN	319	83	379	123	48%
Dueling	524	132	633	172	30%
A3C	293	80	347	94	18%

Table 1: Comparison between the baseline DQN, Dueling and A3C and their NoisyNet version in terms of median and mean human-normalised scores defined in Eq. (18). We report on the last column the percentage improvement on the baseline in terms of median human-normalised score.

3.7 融合上述策略

首先将（改进5:Distributional RL）中的损失函数更换称为（改进4:Multi-step learning），并利用（改进1--Double Q- Learning）计算新的目标值

$$d_t^n = (R_t^n + r_t^n z, p_{\hat{\theta}}(S_{t+n}, a_{t+n}^*))$$

损失函数为

$$D_{KL}(\Phi_z d_t^n || d_t)$$

同时在采样过程中我们通常会减少TD-error，而在本文中我们的损失函数为KL损失，因此我们的（改进2: Prioritized replay）中的优先级定义为

$$p_t \propto (D_{KL}(\Phi_z d_t^n || d_t))^w$$

同时改变（改进3: Dueling networks）由接受期望转向接受价值函数分布，最后更改所有的线性层更换为（改进6: Noisy Nets）

$$p_{\theta}(s, a)^i = \frac{\exp(v_{\eta}^i(\phi) + a_{\psi}^i(\phi, a) - \hat{a}_{\psi}^i(s))}{\sum_j \exp(v_{\eta}^j(\phi) + a_{\psi}^j(\phi, a) - \hat{a}_{\psi}^j(s))}$$

4 实验

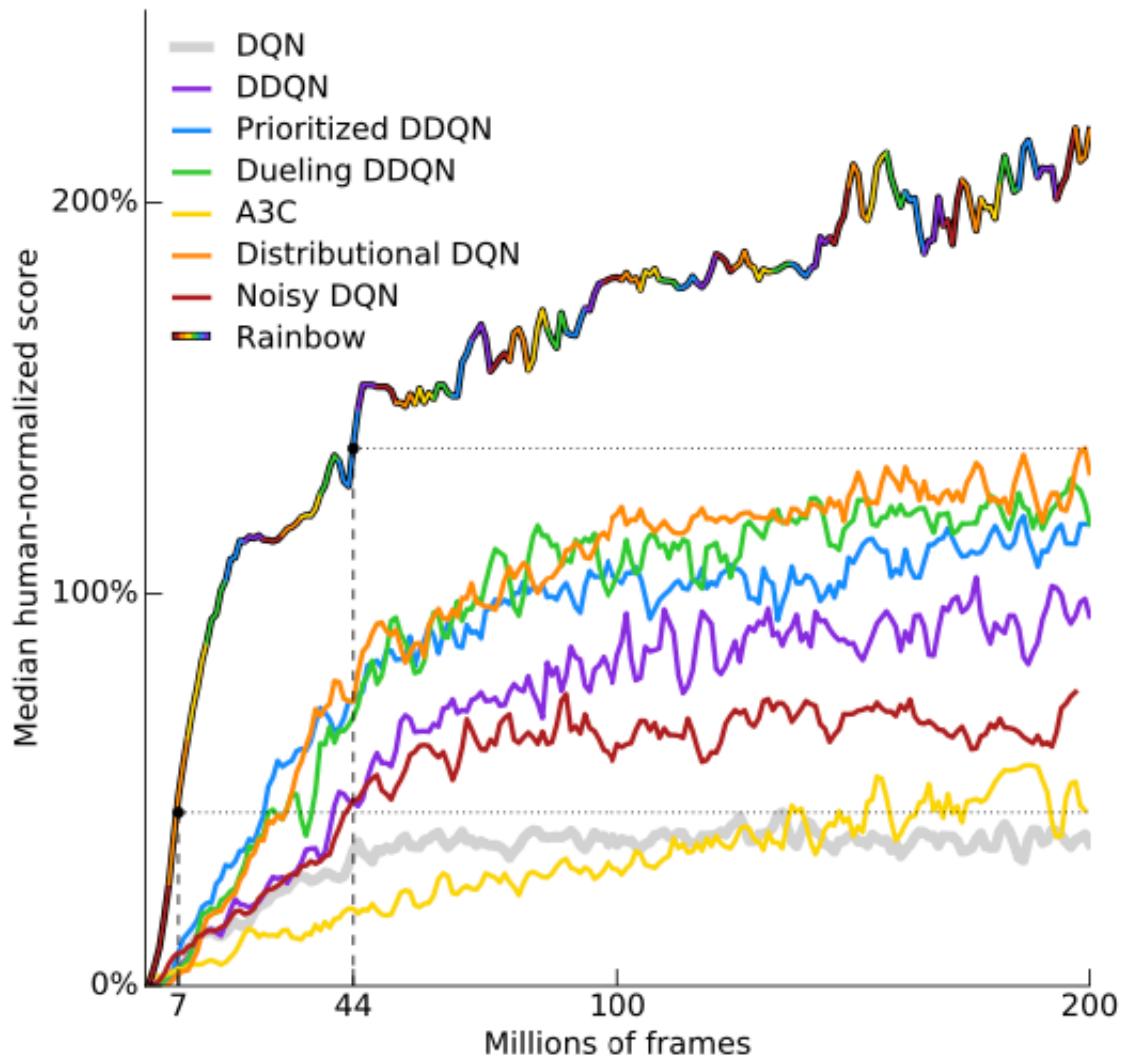


Figure 1: Median human-normalized performance across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN’s best performance after 7M frames, surpass any baseline within 44M frames, and reach substantially improved final performance. Curves are smoothed with a moving average over 5 points.

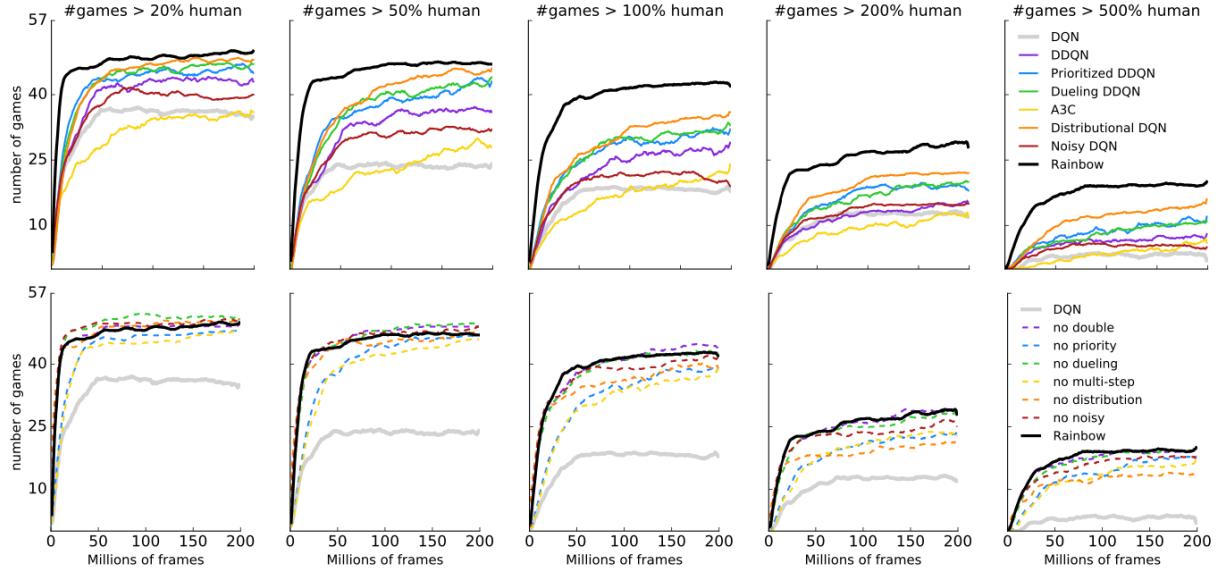


Figure 2: Each plot shows, for several agents, the number of games where they have achieved at least a given fraction of human performance, as a function of time. From left to right we consider the 20%, 50%, 100%, 200% and 500% thresholds. On the first row we compare Rainbow to the baselines. On the second row we compare Rainbow to its ablations.

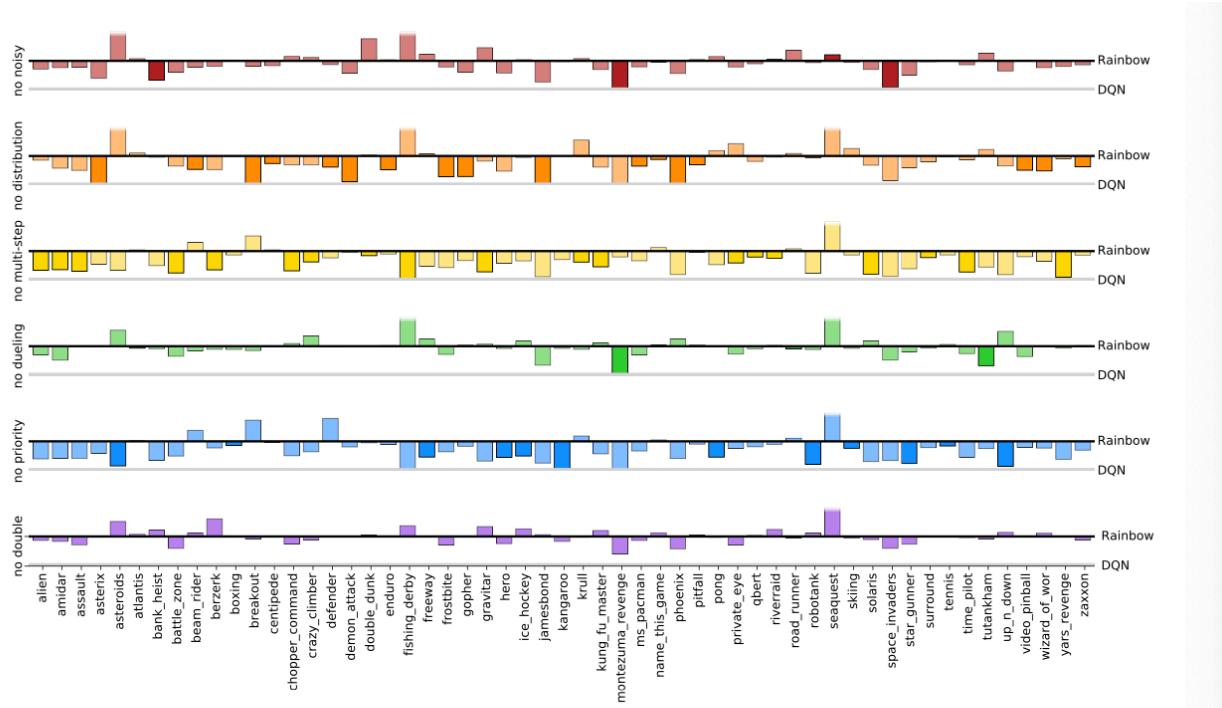


Figure 4: **Performance drops of ablation agents** on all 57 Atari games. Performance is the area under the learning curve, normalized relative to the Rainbow agent and DQN. Two games where DQN outperforms Rainbow are omitted. The ablation leading to the strongest drop is highlighted for each game. The removal of either prioritization or multi-step learning reduces performance across most games, but the contribution of each component varies substantially per game.

5 总结

5.1 结论

1. Rainbow相比较其他现有的算法要更好，速度也会更快
2. 在消融实验中；我们会发现（改进2: Prioritized replay）与（改进4:Multi-step learning）会造成结果中位数大幅度下降；（改进5:Distributional RL）在最开始表现良好，但是最终结果表现较差；同时（改进6: Noisy Nets）通常会有更好的中位数表现，同时由于本次状态中通常是underestimate的，所以（改进1--Double Q- Learning）效果并不显著，（改进3: Dueling networks）提升幅度不大。

5.2 讨论

作者在最后总结他们的工作，主要是从Value- based的Q-learning方法集合中寻找，而没有考虑Purely policy- based的算法（比如TRPO），本文从网络探索、网络初始化、数据使用、损失或函数等方面进行集合，与之相对应的同样有很多工作，未来还可以用很多其他的方法。但是

In general, we believe that exposing the real game to the agent is a promising direction for future research.

5.3 个人感悟

这篇论文的观点很直接，在实际实现的过程中作者做了很多Dirty work，尝试过很多次，并最终证明集成的方式是有效的，以及分析哪些技巧是有效的、哪些技巧是欠佳的，工作量非常大！

[1] [Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. \(2013\). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.](#)

[2] [Van Hasselt, H., Guez, A., & Silver, D. \(2016, March\). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* \(Vol. 30, No. 1\).](#)

[3] [Schaul, T., Quan, J., Antonoglou, I., & Silver, D. \(2015\). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.](#)

[4] [Multi Step Learning](#)

[5] [Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. \(2016, June\). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* \(pp. 1995-2003\). PMLR.](#)

[6] [Bellemare, M. G., Dabney, W., & Munos, R. \(2017, July\). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning* \(pp. 449-458\). PMLR.](#)

[7] [Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., ... & Legg, S. \(2017\). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.](#)

自我介绍

非典型INFJ，一枚普通985的交通信息方向2022级硕士生，主要研究方向为交通数据分析、组合优化等，在学习的路上希望自己可以保持“求知若饥，虚心若愚”。可以在blog.tjdata.site找到我。