

doi:10.1038/nature14236

# Human-level control through deep reinforcement learning

Volodymyr Mnih<sup>1</sup>\*, Koray Kavukcuoglu<sup>1</sup>\*, David Silver<sup>1</sup>\*, Andrei A. Rusu<sup>1</sup>, Joel Veness<sup>1</sup>, Marc G. Bellemare<sup>1</sup>, Alex Graves<sup>1</sup>, Martin Riedmiller<sup>1</sup>, Andreas K. Fidjeland<sup>1</sup>, Georg Ostrovski<sup>1</sup>, Stig Petersen<sup>1</sup>, Charles Beattie<sup>1</sup>, Amir Sadik<sup>1</sup>, Ioannis Antonoglou<sup>1</sup>, Helen King<sup>1</sup>, Dharshan Kumaran<sup>1</sup>, Daan Wierstra<sup>1</sup>, Shane Legg<sup>1</sup> & Demis Hassabis<sup>1</sup>

# 【强化学习算法 1】DQN



张楚珩 🔮

清华大学 交叉信息院博士在读

19 人赞同了该文章

#### 原文传送门:

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." (2015). (ICML版本)

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529. (Nature版本)

**特色**:第一次用强化学习算法从零开始训练玩Atari游戏,能够达到比肩人类玩游戏水平的程度;甚至直接使用video输入也能够有较好的效果。

分类: Model-free、Value-based、Off-policy、Continuous State Space、Discrete Action Space、Support High-dim Input

**理论依据:**Bellman算子有不动点  $q^{r}$ ,反复更新参数使得Q值为作用一次Bellman算子之后的值,Q值会收敛到最优,相应的策略为  $\pi(q) = argmaxQ(q,q)$ 

目标函数:  $L(\theta) = \mathbb{E}_{\theta}[(r_t + \gamma \max_{\mathbf{a}} Q_{\theta}(s_{t+1}, \mathbf{a}) - Q_{\theta}(s_t, s_t))^2]$  , 其中  $\mathbb{E}_{\theta}$  代表off-policy采样,  $\mathbb{E}_{\epsilon}$  代表on-policy采样。

更新公式:  $\theta \leftarrow \theta - \alpha(r_t + \gamma \max_a Q_\theta(s_{t+1}, a) - Q_\theta(s_t, a_t)) \nabla_\theta Q_\theta(s_t, a_t)$ 

算法:

#### 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 D
         Sample random minibatch of transitions (\phi_j, a_j, r_j, \phi_{j+1}) from \mathcal{D}
                                                               for terminal \phi_{j+1}
                      r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta)
                                                               for non-terminal \phi_{j+1}
         Perform a gradient descent step on (y_i - Q(\phi_i, a_i; \theta))^2 according to equation 3
    end for
end for
```

## 改进:

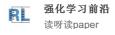
- 1. Target Network: 在Nature版本的文章里面就加入了一个目标网络,这个网络每隔一些episode与最新的网络同步一下权重,这样更新较慢的网络用于查询max步的Q值。
  - $\theta \leftarrow \theta \alpha(r_t + \gamma \max_{a} Q_{\theta^-}(s_{t+1}, a) Q_{\theta}(s_t, a_t)) \nabla_{\theta} Q_{\theta}(s_t, a_t)$  ,其中  $_{\theta^-}$  就是target network的权重。
- 2. <u>Double Q-Learning</u>:如果Q函数估值不准,那么每次取max会引起高估,因此用两个Q网络来解决这个问题,一个网络选择最优的action,另一个网络对其估值。  $X_i^{Double} = r_i + \gamma Q_i(e_{i+1}, arg max} Q_i(e_{i+1}, a))$ 。它应用到DQN的时候可以直接把target network当做第二个Q网络,即  $g_{T \to \theta}$ 。
- 3. Prioritized Experience Replay: 希望每次采样是对于更新帮助比较大的样本,认为该样本上一次在Q网络上的的TD error的绝对值越大越有用,因此有priority p<sub>1</sub> = |r<sub>1</sub> + γQ<sub>r</sub> (s<sub>1+1</sub>, orgmax Q<sub>0</sub> (s<sub>1+1</sub>, a)) Q<sub>0</sub> (s<sub>1</sub>, a<sub>2</sub>)],每次采样的概率为 【【】 】 ,并且加上权重 w<sub>1</sub> = (NP<sub>1</sub>)<sup>-9</sup> 确保有采样之后也是无偏的。
- 4. <u>Dueling Network</u>: DQN里面的Q网络一般是输出一个 <sub>Al</sub> 维的向量,每一位代表一种action的Q值,现在把它拆分成 <sub>Al+1</sub> 维的向量,分别代表V(state value function)和A(advantage)。好处是某个state上采取不同action共有价值的部分能更好地被估计。

编辑于 2018-09-19



▲ 赞同 19 ▼ 11 条评论 ▼ 分享 ♥ 喜欢 ★ 收藏 …

## 文章被以下专栏收录



进入专栏