强化学习

第九讲：深度强化学习

教师：赵冬斌　朱圆恒　张启超

中国科学院大学
中国科学院自动化研究所

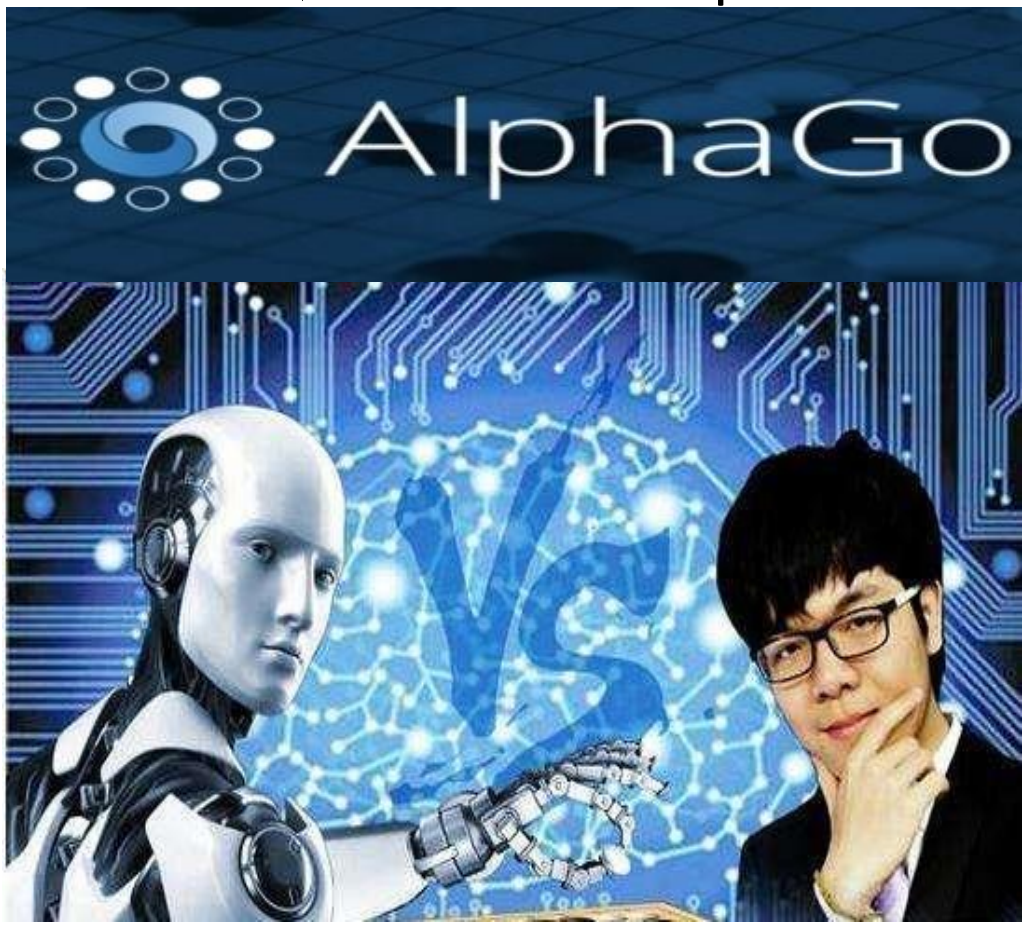
April 15, 2019

课程目录

深度强化学习的发展

- 讲到深度强化学习会想到的是?

2016年李世石 VS AlphaGo



深度强化学习的发展

- 单智能体感知决策

2013 NIPS，DeepMind提出Atari视频游戏的深度强化学习算法DQN，得分超过人类水平

神经信息处理系统大会 (Conference and Workshop on Neural Information Processing Systems) CCF-A类会议

2013-2015 → **2016-2017** → **2017-2018** → **2019-2020**

2015 Nature，DeepMind提出Atari视频游戏的深度强化学习算法DQN，得分超过人类高级水平

深度强化学习的发展

- 完全信息零和博弈

2016 Nature，DeepMind的AlphaGo以4:1的大比分战胜了世界围棋顶级选手李世石

2013-2015 → 2016-2017 → 2017-2018 → 2019-2020

- 蒙特卡洛树搜索
- Actor-Critic
- 自我博弈

2017 Nature，DeepMind的AlphaGo Zero不依赖人类的棋谱数据，自我博弈学习达到人类顶级水平

深度强化学习的发展

- 非完全信息双人零和博弈
  - 蒙特卡洛搜索
  - 虚拟遗憾最小化算法（CFR）

…

2017 Science, 加拿大阿尔伯特大学开发的DeepStack，世界上第一个在“1对1无限注德州扑克”上击败了职业扑克玩家的计算机程序

2013-2015 → **2016-2017** → **2017-2018** → **2019-2020**

2017-2018, 卡内基梅隆大学开发的Libratus，1V1德扑冷扑大师系列人机大战，先后取胜，获得NIPS 2017最佳论文

深度强化学习的发展

- 非完全信息多人混合博弈

2019 Science, 谷歌第一视角多个体合作游戏，雷神之锤

2019 Science, CMU的六人桌德扑Bot Pluribus

2019.5, OpenAI Five，Dota2的人机大战

2013-2015 → **2016-2017** → **2017-2018** → **2019-2020**

- 多智能体深度强化学习
- 递归神经网络

…

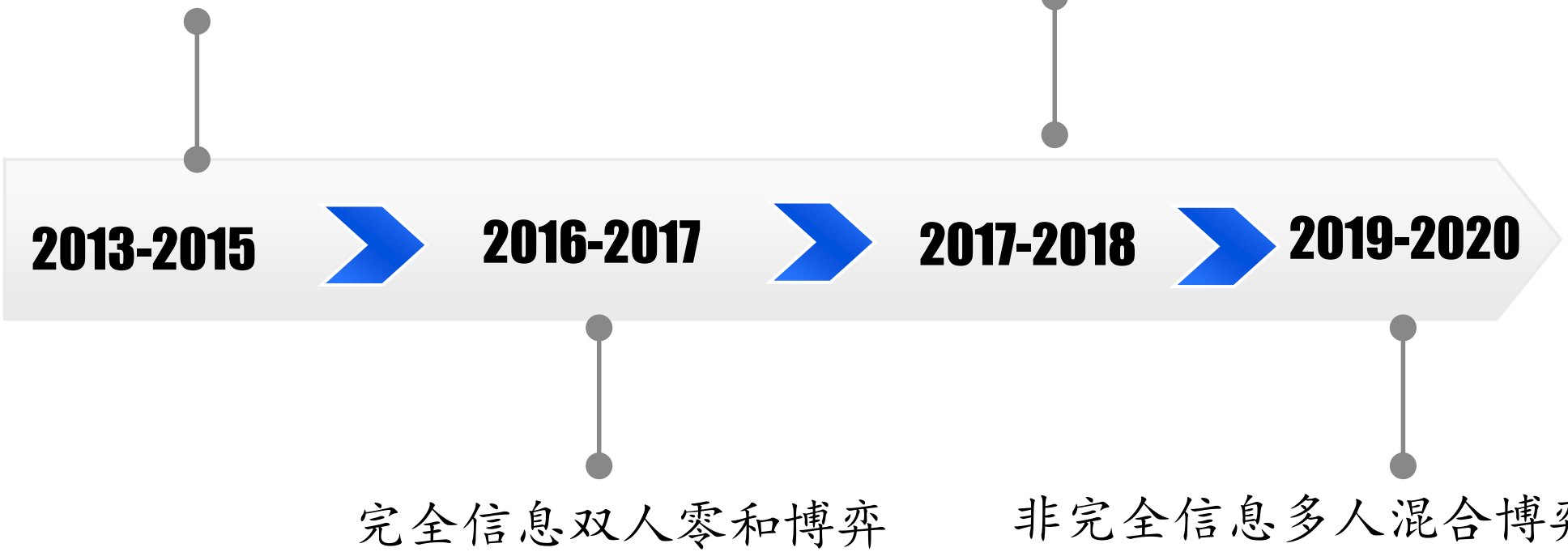
2019.8,微软麻将AI Suphx, 10段

2019.11 Nature，谷歌Alpha Star，星际争霸II达到大师级水平

2019.12，腾讯绝悟AI击败王者荣耀顶尖职业玩家

深度强化学习的发展

深度强化学习近年来的发展趋势



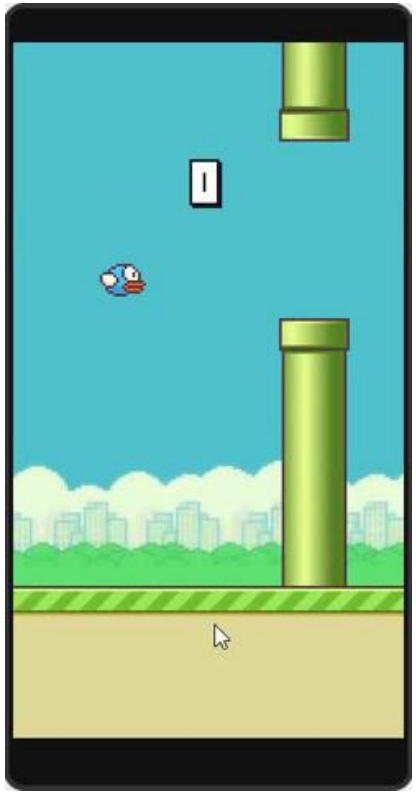
有很多算法及应用方面的问题还有待大家去探索解决

深度强化学习的定义

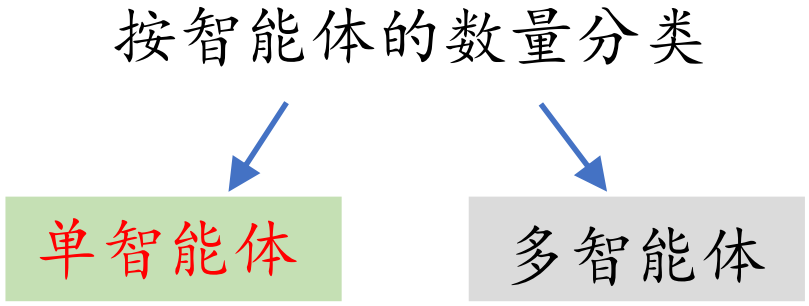
深度强化学习是指什么？

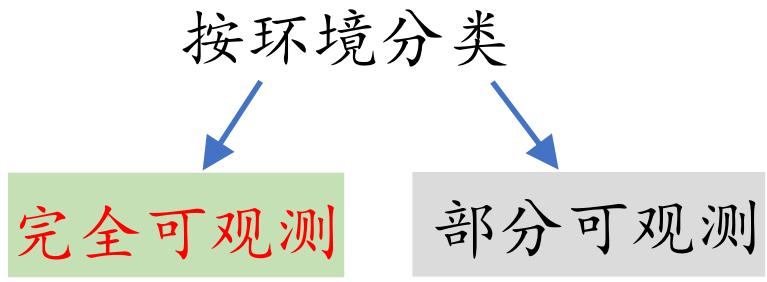
高维输入状态到策略的映射模型

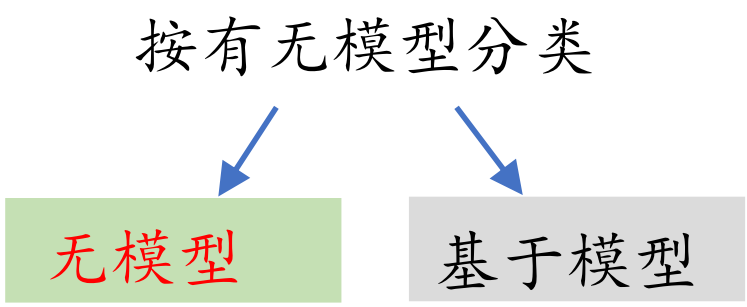
- 字面理解：深度学习+强化学习 = 深度强化学习
- 本质上：使用深度神经网络作为强化学习的函数近似器
- 针对复杂高维原始数据(图像、视频等)输入的决策问题，深度学习强大的特征提取能力，可将原始数据表征为和问题相关的特征

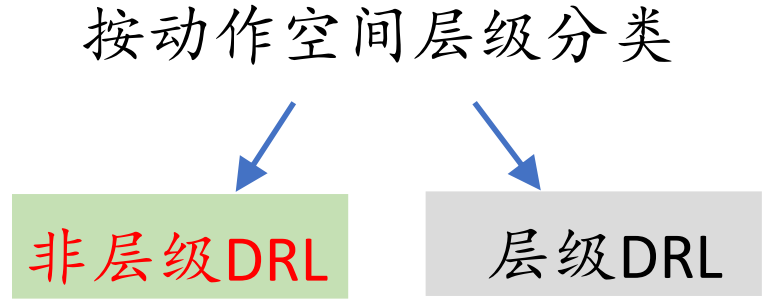


深度强化学习的分类



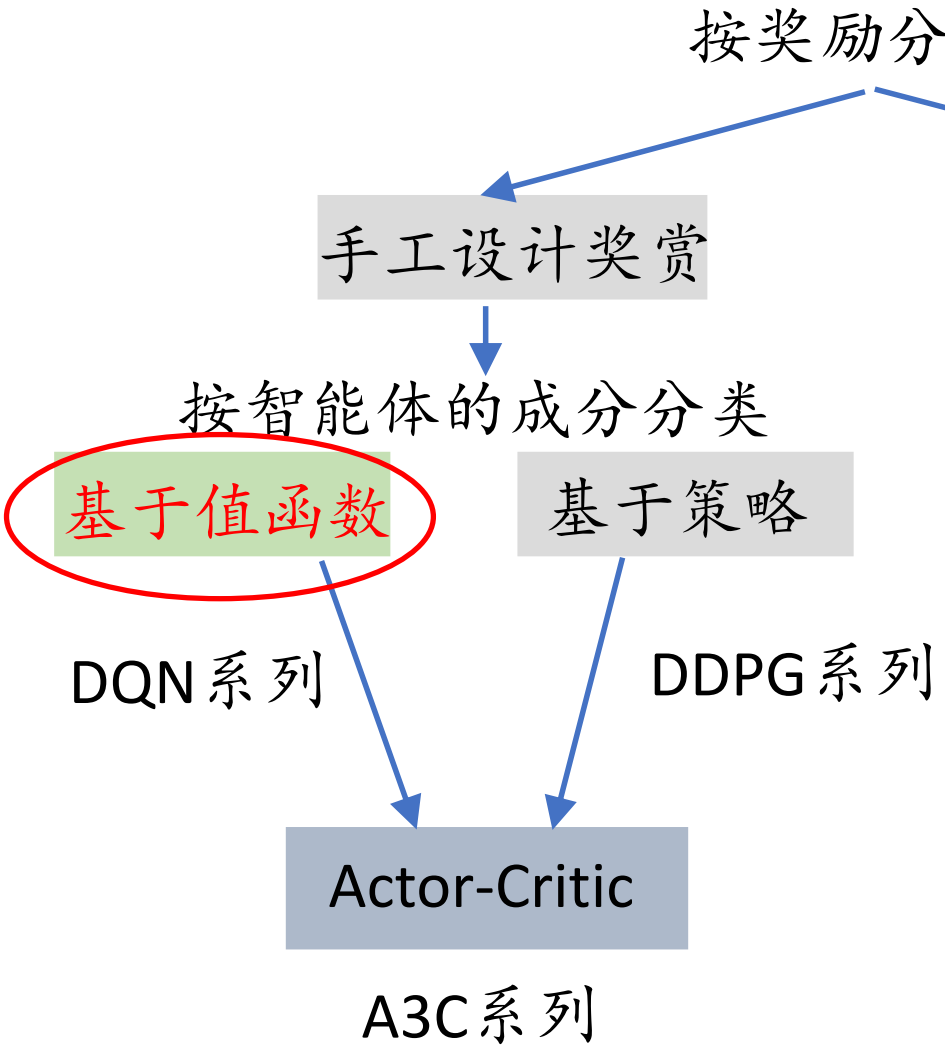






课程介绍的内容均是针对完全可观测环境下的单智能体无模型DRL方法

深度强化学习的分类



- 经验回放
- 目标网络
- 重要性采样
- …

课程目录

基于Q函数的深度强化学习

学习目标

- 基于Q函数的深度强化学习

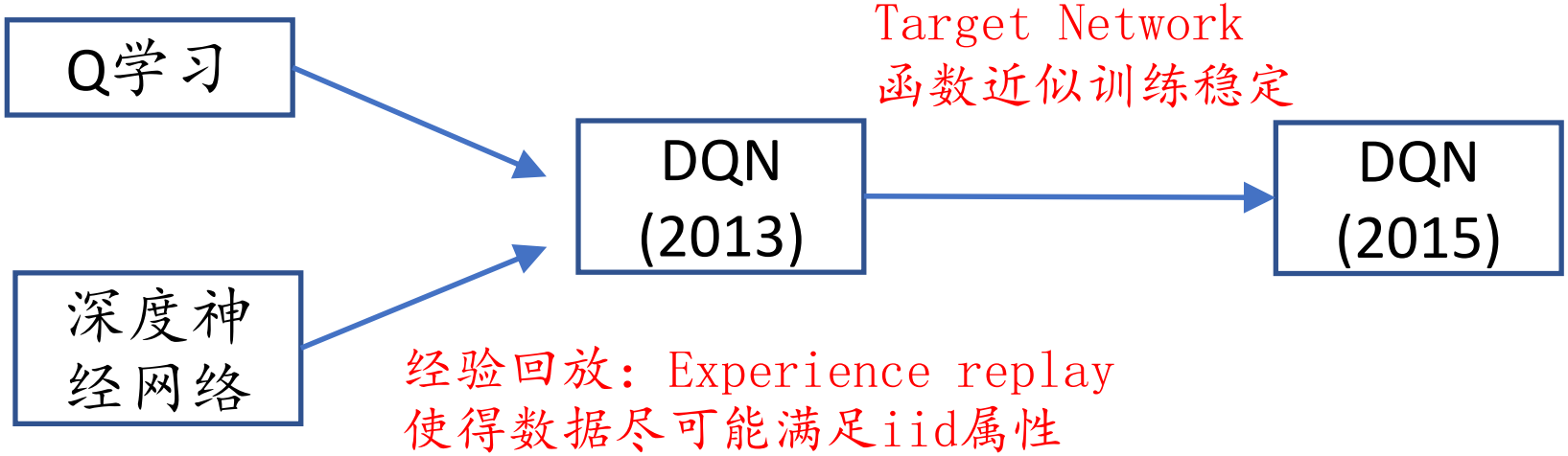
1. DQN(Deep Q Learning)
2. Double DQN
3. Prioritized Experience Replay
4. Dueling DQN

- 如何将深度网络与Q学习结合？
- 离散空间的几类深度Q学习的改进方法

理解如何基于深度网络的复杂函数逼近器应用Q学习

参考：CS 285: Deep Reinforcement Learning, Decision Making, and Control, Sergey Levine

DQN (Deep Q Learning)



- 首次将最火热的深度神经网络与强化学习结合，解决复杂高维输入的决策控制问题，确保训练过程的稳定性
- 证明了能够通过raw pixels 解决游戏问题
- 可适用于绝大多数的Atrai游戏

1. Playing Atari with Deep Reinforcement Learning (NIPS2013)
2. Human-level control through deep reinforcement learning (Nature2015)

DQN (Deep Q Learning)

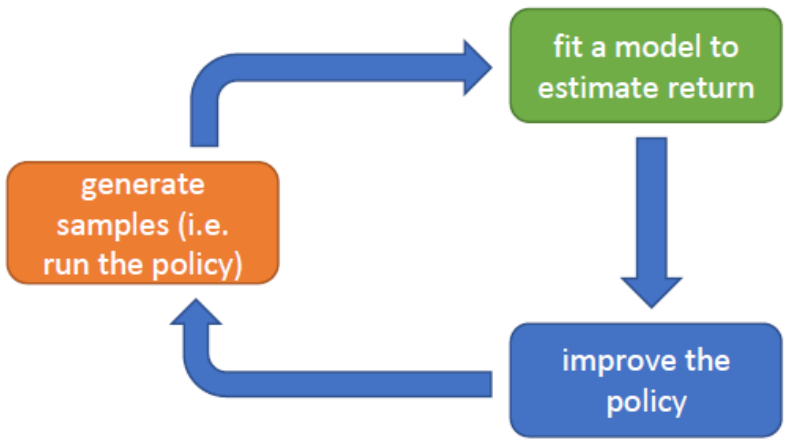
- Q学习回顾

传统Q学习步骤：

1. 利用某些策略收集数据集 $\{(s_i, a_i, s_i', r_i)\}$
2. 估计Q值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a_i'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \arg\min_\varphi \frac{1}{2}\Sigma_i \|Q_\varphi(s_i,a_i) - y_i\|$

N

$$Q_\phi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$$



$$\mathbf{a} = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$$

DQN (Deep Q Learning)

- 当Q学习遇见深度神经网络

传统在线Q学习步骤：

N

1. 初始执行动作a_i，得到观测数据 (s_i, a_i, s_i', r_i)
2. 估计Q值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \varphi - \alpha \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - y_i)$

- 每一次只用1个样本训练网络，存在很大方差，Batchsize=1
- 样本存在相关性
- 算法很难保证收敛性

DQN (Deep Q Learning)

- 当Q学习遇见深度神经网络

问题1：样本不满足iid条件，时间相关性很强

在线Q学习步骤：

1. 初始执行动作a_i，得到观测数据$\{(s_i, a_i, s_i', r_i)\}$
2. 估计Q目标值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \varphi - \alpha \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - y_i)$

问题2：

■ Q学习理论上无法证明收敛，因为更新并非是梯度下降

$y_i \leftarrow r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$

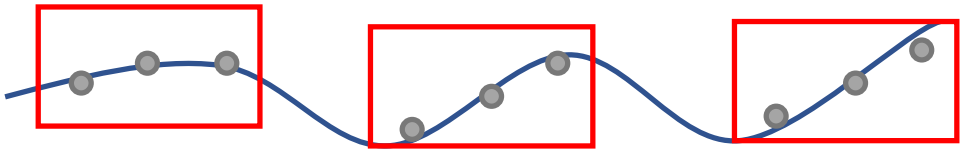
目标值同样依赖参数φ，一直在变化；

计算梯度时却忽略了该项；

DQN (Deep Q Learning)

在线Q学习步骤：

1. 初始执行动作a_i，得到观测数据 $\{(s_i, a_i, s_i', r_i)\}$
2. 估计Q目标值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \varphi - \alpha \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - y_i)$

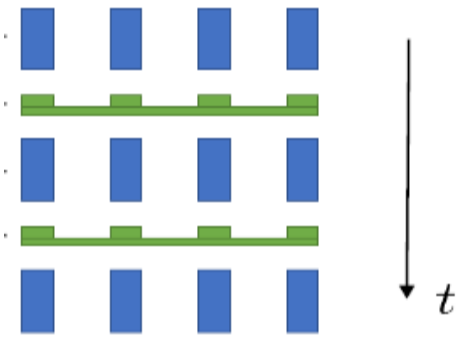


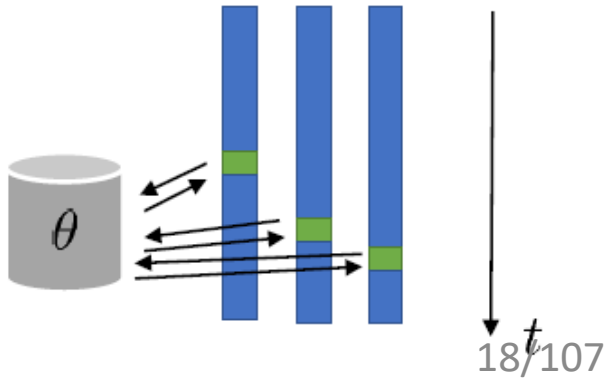
过拟合到局部数据

同步并行-on policy

异步并行

收集 $\{(s_i, a_i, s_i', r_i)\}$
更新参数 φ
收集 $\{(s_i, a_i, s_i', r_i)\}$
更新参数 φ





DQN (Deep Q Learning)

在线Q学习步骤：

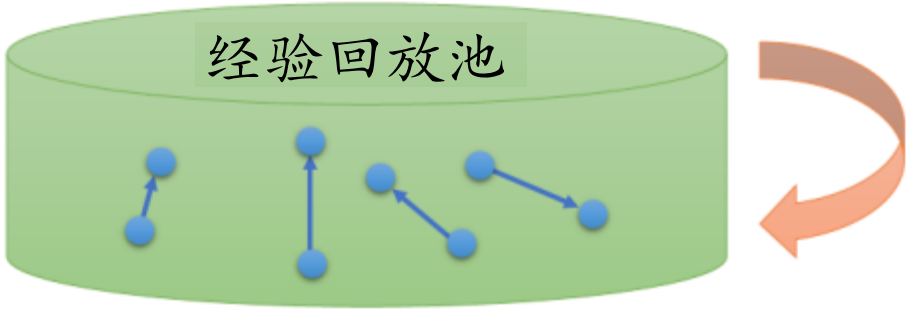
1. 初始执行动作a_i，得到观测数据 $\{(s_i, a_i, s_i', r_i)\}$
2. 估计Q目标值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \varphi - \alpha \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - y_i)$

传统Q学习步骤：

1. [redacted]

N

2. 估计Q值 $y_i \leftarrow r(s_i, a_i) + \gamma max_{a_i'} Q_\varphi(s_i', a_i')$
3. 更新参数 $\varphi \leftarrow \arg\min_\varphi \frac{1}{2} \Sigma_i \| Q_\varphi(s_i, a_i) - y_i \|$



DQN (Deep Q Learning)

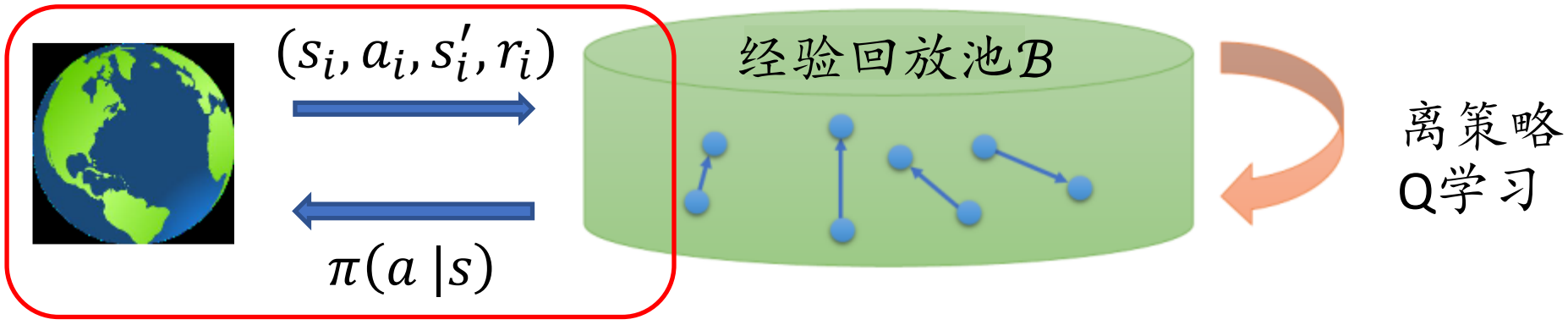
经验回放下的Q学习步骤：

1. 在经验池 $\mathcal{B}$ 采样batch数据 $\{(s_i, a_i, s_i', r_i)\}$
2. 更新参数 $\varphi \leftarrow \varphi - \alpha \sum_i \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - [r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')])$

- 样本相关性大幅降低
- Batch训练降低方差

- 如何得到经验池数据？
  周期性存储并更新经验池数据

利用历史策略收集样本，存入样本回放池，执行2-3步迭代



DQN (Deep Q Learning)

经验回放下的Q学习步骤：

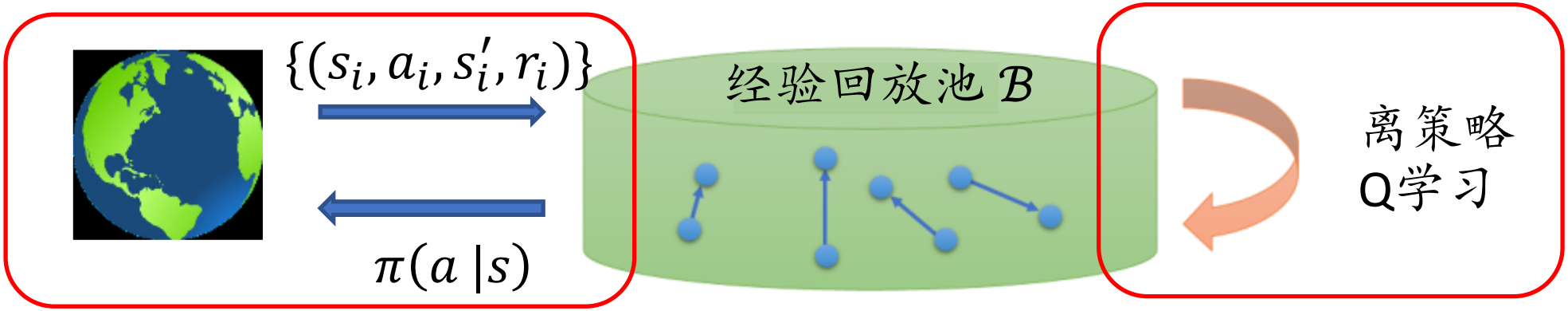
1. 利用某些策略收集样本 $\{(s_i, a_i, s_i', r_i)\}$，加入样本池 $\mathcal{B}$

N

2.在经验池 $\mathcal{B}$ 采样batch数据

3. 更新参数 $\varphi \leftarrow \varphi - \alpha \sum_i \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - [r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')])$

通过经验回放缓解了数据iid的问题



这就是2013年DQN最初的结构

DQN (Deep Q Learning)

经验回放下的Q学习步骤：

1. 利用某些策略收集样本 $\{(s_i, a_i, s_i', r_i)\}$，加入样本池 $\mathcal{B}$

N

2. 在经验池 $\mathcal{B}$ 采样batch数据

3. 更新参数 $\varphi \leftarrow \varphi - \alpha \sum_i \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - [r(s_i, a_i) + \gamma max_{a'} Q_\varphi(s_i', a_i')$

没有目标值中的梯度传递，
而目标值又一直在变化

导致DQN的训练不稳定

DQN (Deep Q Learning)

经验回放+目标网络结合的Q学习步骤：

1. 保存目标Q网络参数$\varphi' \leftarrow \varphi$

N

2.利用某些策略收集样本 $\{(s_i, a_i, s_i', r_i)\}$，加入样本池$\mathcal{B}$

K

3.在经验池$\mathcal{B}$ 采样batch数据

4. 更新参数$\varphi \leftarrow \varphi - \alpha \sum_i \frac{dQ_\varphi}{d\varphi}(s_i, a_i)(Q_\varphi(s_i, a_i) - [r(s_i, a_i) + \gamma max_{a'} Q_{\varphi'}(s_i', a_i')])$

内层类似于一个回归任务

内层迭代的时候目标值不再发生改变

DQN (Deep Q Learning)

经典DQN算法：

1. 利用ε-greedy策略执行动作a_i，收集样本$\{(s_i, a_i, s_i', r_i)\}$，加入经验池$\mathcal{B}$

2.在经验池$\mathcal{B}$ 中采样batch数据$\{(s_j, a_j, s_j', r_j)\}$

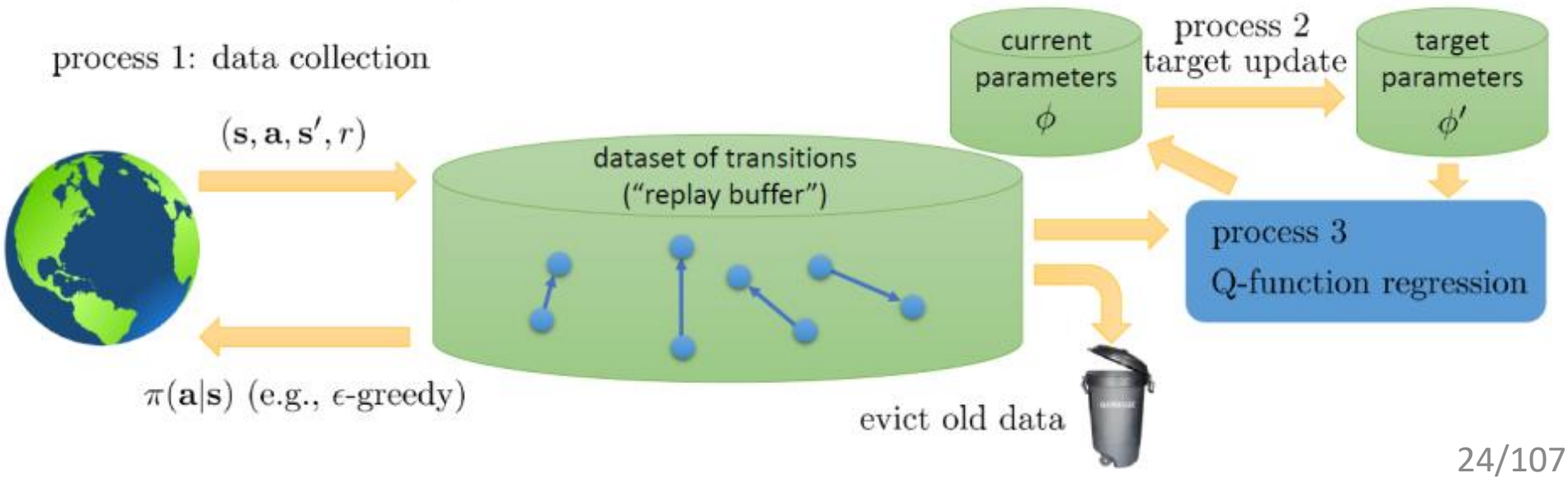
3. 计算目标网络估计值 $y_j \leftarrow \begin{cases} y_j \leftarrow r_j ; j\text{为终止时刻} \\ r(s_j,a_j)+\gamma max_{a'} Q_{\varphi'}(s_j', a_j') \end{cases}$

4. 更新Q网络参数 $\varphi \leftarrow \varphi - \alpha \sum_j \frac{dQ_\varphi}{d\varphi}(s_j,a_j)(Q_\varphi(s_j,a_j) - y_j)$

5. 更新目标网络参数 $\varphi' \leftarrow \varphi$

K=1

N=1



DQN (Deep Q Learning)

这就是2015年DQN的伪代码

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-}=\theta$

For episode $=1, M$ **do**

  Initialize sequence $s_1=\{x_1\}$ and preprocessed sequence $\phi_1=\phi(s_1)$

  For $t=1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t=\text{argmax}_a Q(\phi(s_t),a;\theta)$

    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 D

    Set $y_j=\begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j+\gamma \max_{a'} \hat{Q}(\phi_{j+1},a';\theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j-Q(\phi_j,a_j;\theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q}=Q$

  End For

End For

初始化

步骤1

步骤2-3

步骤4-5

DQN-多步回报

- DQN的多步回报版本

- 标准DQN $y_j \leftarrow r(s_j, a_j) + \gamma Q_{\varphi'}(s'_j, \text{argmax} Q_{\varphi'}(s'_j, a'_j))$

如果$Q_{\varphi'}$逼近器很差，
r的好坏影响会很大

如果$Q_{\varphi'}$逼近器很好，
s'_j和a'_j的值又非常关键

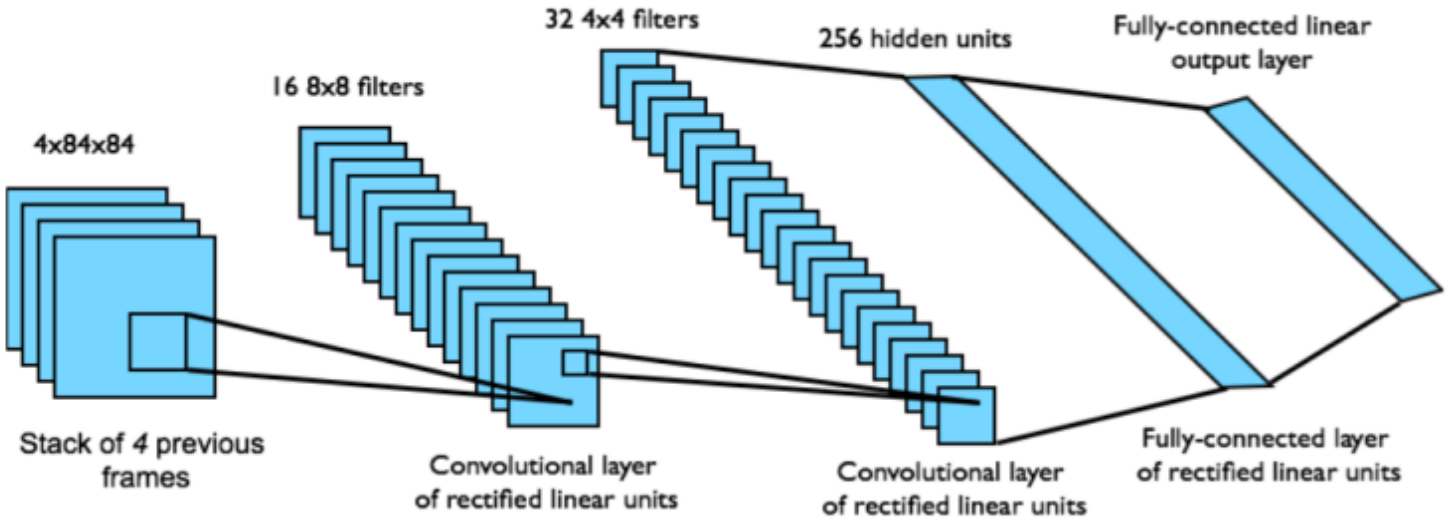
想要得到更低的方差，N步回报估计器

$$y_{j,t} \leftarrow \sum_{t'=t}^{t+N-1} \gamma^{t-t'} r(s_{j,t'}, a_{j,t'}) + \gamma^N \max_{a_{j,t+N}} Q_{\varphi'}(s'_{j,t+N}, a'_{j,t+N}))$$

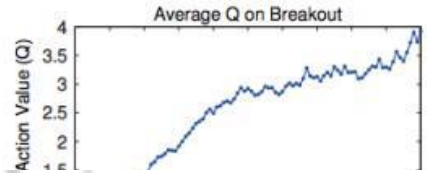
DQN (Deep Q Learning)

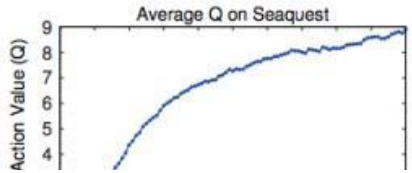
DQN in Atari

- 以图片s为状态的端到端学习(估计Q(s,a))
- 输入状态s是最近4帧的原始像素图像
- 输出为离散动作空间对应的Q(s,a)
- 奖赏值为每次的得分

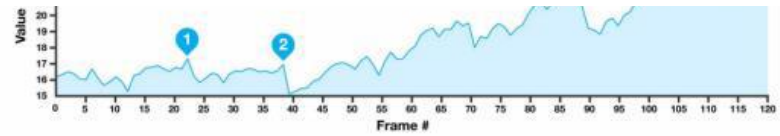


DQN (Deep Q Learning)





Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0



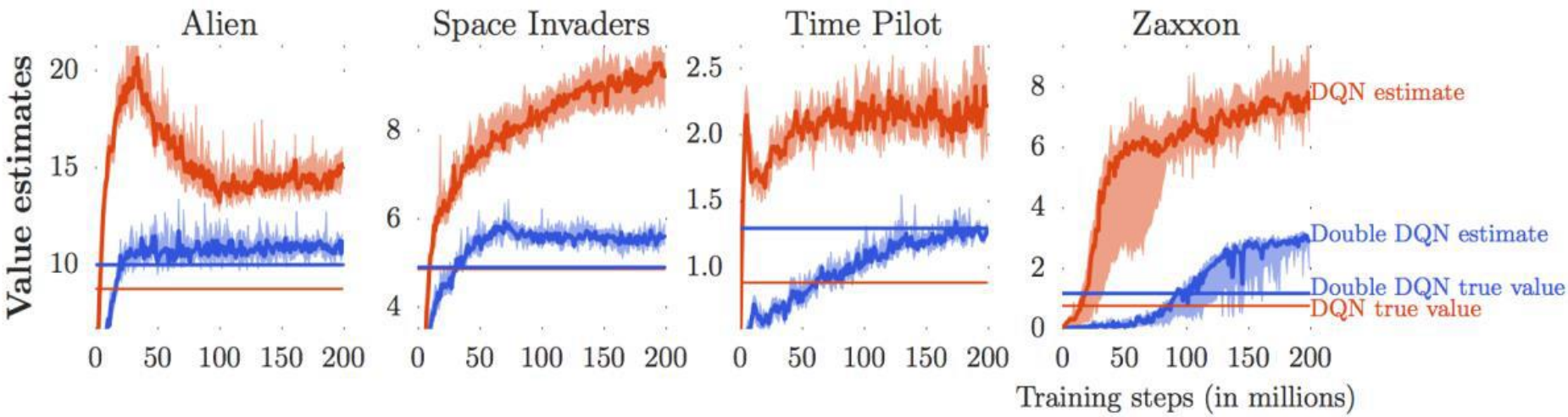
问题思考：1.Q值估计是否准确？

2. 经验池均匀采样的效率是否好？

源码：sites.google.com/a/deepmind.com/dqn/

DQN (Deep Q Learning)

- 问题1：DQN中Q值估计的准确么？



答案是：Q值估计并不准确

Double DQN

- DQN中的过估计问题

目标网络估计值：$y_i \leftarrow r(s_i, a_i) + \gamma max_{a_i'} Q_{\varphi'}(s_i', a_i')$

$$max_{a_i'} Q_{\varphi'}(s_i', a_i') = Q_{\varphi'}\left(s_i', \underset{a_i'}{\text{argmax}}\, Q_{\varphi'}(s_i', a_i')\right)$$

目标Q网络值估计　动作的选择

- TD目标值中的max操作，将引入一个正向的偏差，导致下一时刻的目标值存在过估计

假设存在两个随机变量X_1, X_2：

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

X_1的期望为0.5　X_2的期望为1，左项的期望不小于1

Double DQN

- DQN中的过估计问题

$$max_{a_i'} Q_{\varphi'}(s_i', a_i') = Q_{\varphi'}\left(s_i', \underset{a_i'}{\operatorname{argmax}}\, Q_{\varphi'}(s_i', a_i')\right)$$

目标网络值估计　　动作的选择

如果这两项不再相关，将大大减轻过估计问题

解决思路：使用不同的网络来分别计算目标Q网络值和选择动作

- Double DQN：利用两个网络

$$Q_{\varphi_A}(s, a) \leftarrow r(s, a) + \gamma Q_{\varphi_B}\left(s', \underset{a_i'}{\operatorname{argmax}} Q_{\varphi_A}(s', a')\right)$$

$$Q_{\varphi_B}(s, a) \leftarrow r(s, a) + \gamma Q_{\varphi_A}\left(s', \underset{a_i'}{\operatorname{argmax}} Q_{\varphi_B}(s', a')\right)$$

Double DQN

- DQN中的过估计问题

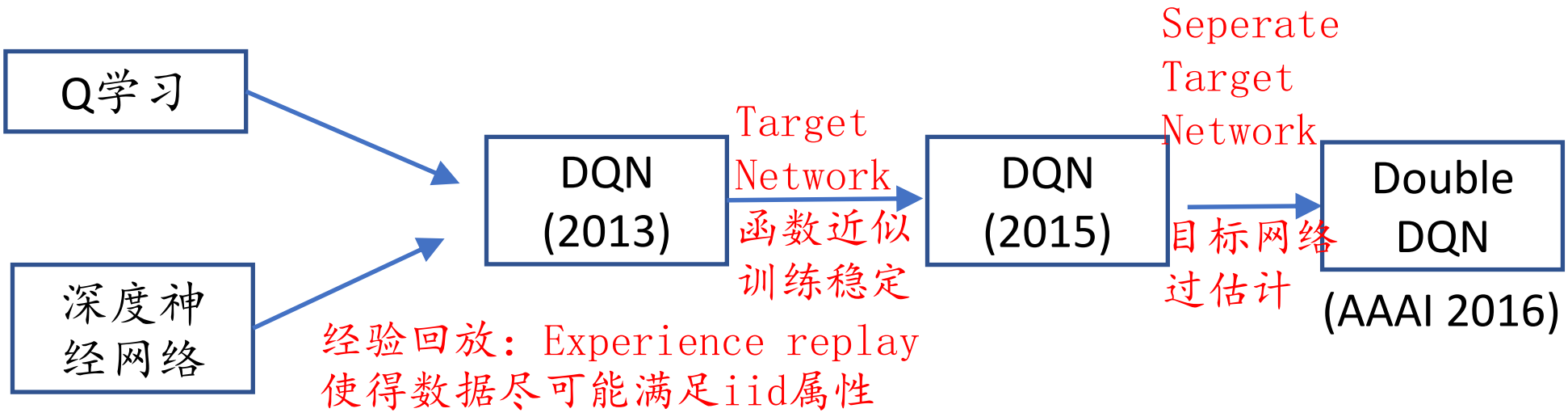
  如何利用DQN得到两个Q网络呢？

  思路：使用当前Q网络Q_φ和目标Q网络$Q_{\varphi'}$

  - 标准DQN $y_j \leftarrow r(s_j, a_j) + \gamma Q_{\varphi'}(s'_j, \text{argmax} Q_{\varphi'}(s'_i, a'_i))$
  - Double DQN $y_j \leftarrow r(s_j, a_j) + \gamma Q_{\varphi'}(s'_j, \text{argmax} Q_{\varphi}(s'_i, a'_i))$

  ■ 利用当前Q网络来选择动作；
  ■ 利用目标Q网络值来估计目标值

Double DQN



Algorithm 1 Double Q-learning

Initialize Q^A, Q^B, s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

- Q_{φ_A}和Q_{φ_B}两个网络是相对独立的，一般不同一时刻更新
- 每次随机选择一个网络更新

Prioritized Experience Replay (ICLR 2016)

- 问题2： 经验池均匀采样的效率是否足够好？

  - DQN 算法的一个重要改进是Experience Replay
  - 训练时从经验池中均匀采样

Prioritized Experience Replay 就是维护了一个带优先级的经验回放

- 每个样本的价值不同，采样的优先级应该不同
- 利用TD 误差去衡量优先级，TD误差大样本价值高，从而剔除价值低的样本
- 经验池的样本保持“先入先出”原则

TD 误差： $|r(s_i,a_i)+\gamma max_{a_i'}Q_{\varphi'}(s_i',a_i')-Q_{\varphi}(s,a)|$

Prioritized Experience Replay (ICLR 2016)

优先级经验回放带来的问题：

- TD 误差对噪声敏感
- TD 误差小的样本长时间得不到不更新
- 过分关注TD 误差大的样本，丧失了样本多样性

- 使用某种分布采样样本(优先级高的样本被长期选择)，会引入偏差Bias

Prioritized Experience Replay (ICLR 2016)

优先级确定与根据优先级采样

（1）优先级确定方法（TD误差优先化）

如： $p_i = |\delta_i| + \epsilon$

通过加入一个小的噪音项，增加一些多样性，确保非零

（2）根据优先级采样

随机优先化：stochastic prioritization:

采样概率 α--- TD误差的重要性

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

α参数的选择决定了优先级的程度，α=0为标准DQN

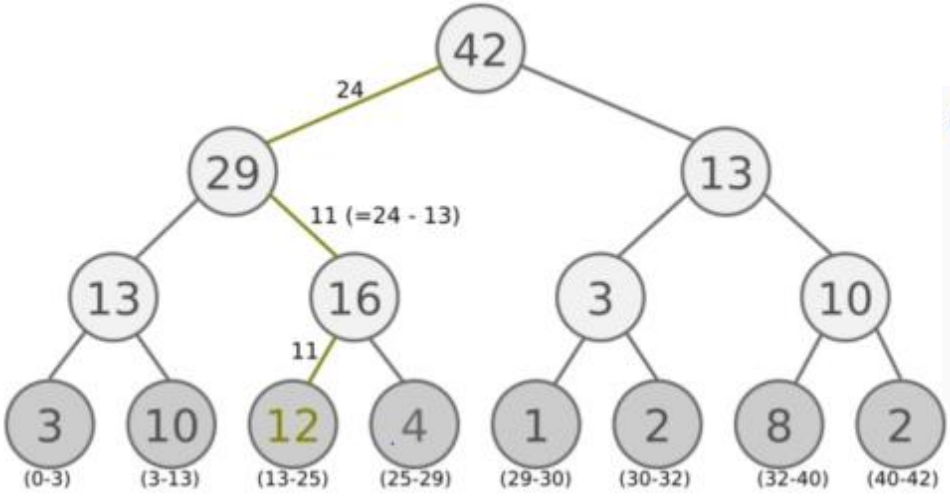
Prioritized Experience Replay (ICLR 2016)

每一次都需要更新经验池中所有样本的优先级，遍历整个经验池选择优先值高的样本，计算量较大的问题

存储、采样方式：Sum-Tree

[0-7] [7-14] [14-21]
[21-28] [28-35] [35-42]

就将根节点的总的优先值除以batch_size，划分为batch_size个区间




```
def retrieve(n, s):
  if n is leaf_node: return n

  if n.left.val >= s: return retrieve(n.left, s)
  else: return retrieve(n.right, s - n.left.val)
```

Prioritized Experience Replay (ICLR 2016)

优先级高的样本具有更高利用率带来的“有偏”

引入重要性采样权重来平衡“有偏”问题

重要性采样退火 $w(i) = \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta}$

一般需要归一化

$$\Delta \leftarrow \Delta + w(i)\delta_i \nabla_\theta Q(s_{i-1}, a_{i-1})$$

$\beta: 0 \rightarrow 1$

前期注重优先级高的样本的利用率
后期注重无偏性

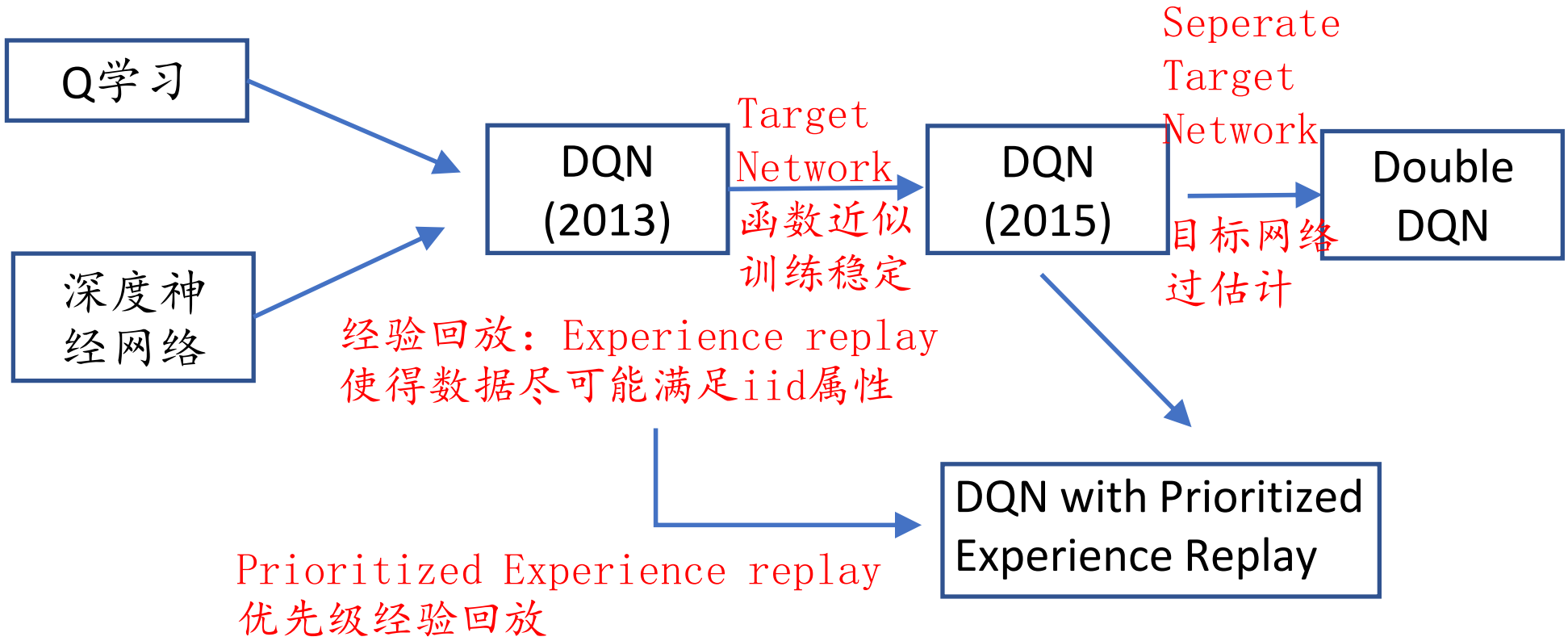
经典的强化学习的场景下，更新的无偏性是训练最后接近收敛最重要的部分

Prioritized Experience Replay (ICLR 2016)

Algorithm 1 Double DQN with proportional prioritization

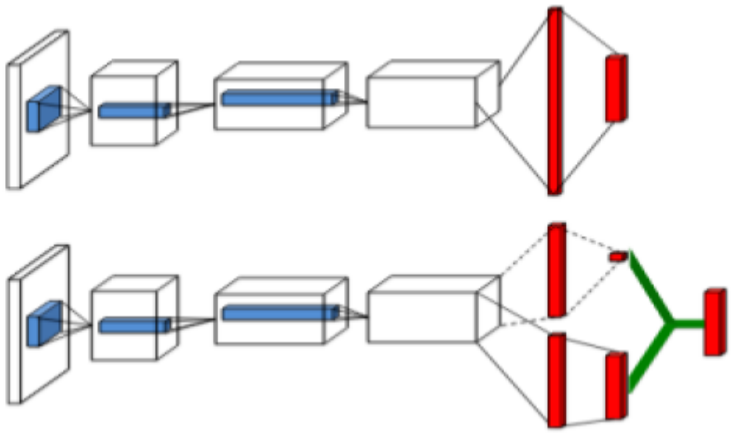
Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ 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 (ICLR 2016)



Dueling-DQN(ICML 2016)

- 将Q 函数分解成V 函数和优势(A) 函数



•特征提取层参数共享，
•在DQN全连接层将网络输出一分为二

$$Q(s,a) = V(s) + A(s,a)$$

- 不依赖动作的值函数$V(s)$
- 依赖动作的优势函数$A(s,a)$

- 优势函数的定义为：

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

Dueling-DQN(ICML 2016)

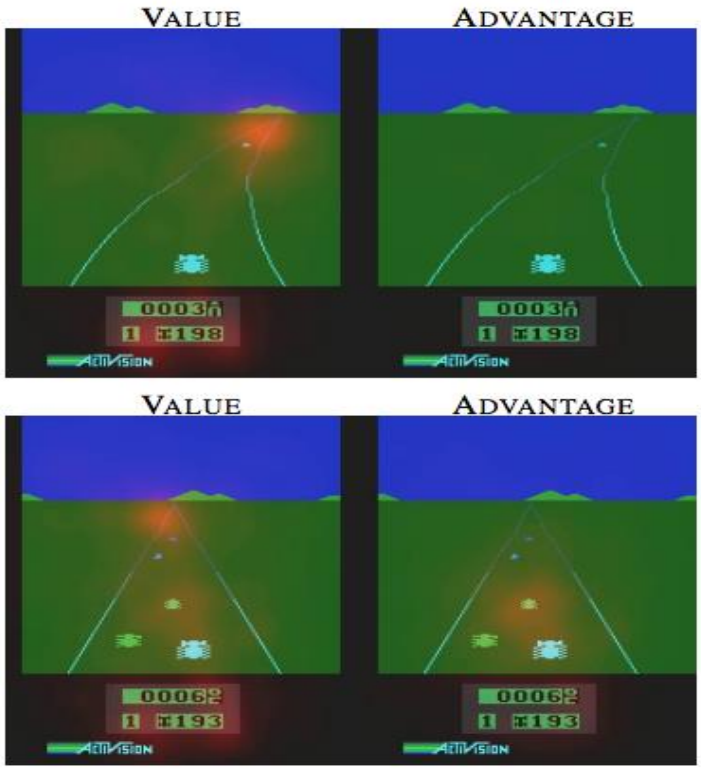
- 为什么将Q网络分解成动作依赖和动作不依赖的两个网络

  - 对于很多状态并不需要估计每个动作的值，增加了V函数的学习机会
  - V函数的泛化性能好，当有新动作加入时，并不需要重新学习
  - 减少了Q 函数由于状态和动作维度差导致的噪声和突变

Dueling-DQN(ICML 2016)

值函数：当前方没有车辆时，重点关注道路状态，动作变化对Q值影响不大，应该着重学习V函数

优势函数：当周围车辆密集时，动作变化对Q值影响增大，次数优势函数的学习可有助于估计更准确的Q函数



Dueling-DQN(ICML 2016)

原始公式：

$$Q(s,a) = V(s) + A(s,a)$$

改进版本：

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

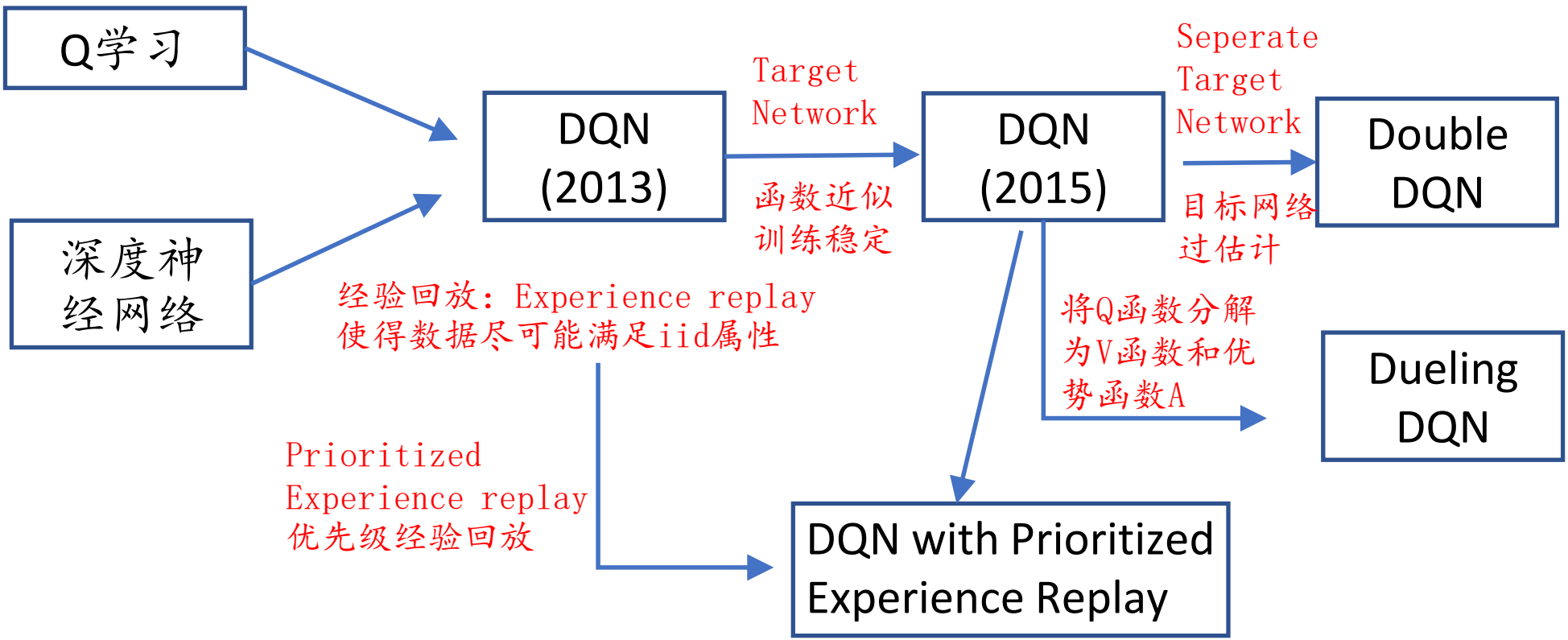
$$a^* = \arg\max_{a' \in \mathcal{A}} Q(s,a';\theta,\alpha,\beta)$$

$$Q(s,a^*;\theta,\alpha,\beta) = V(s;\theta,\beta)$$

平均值版本：

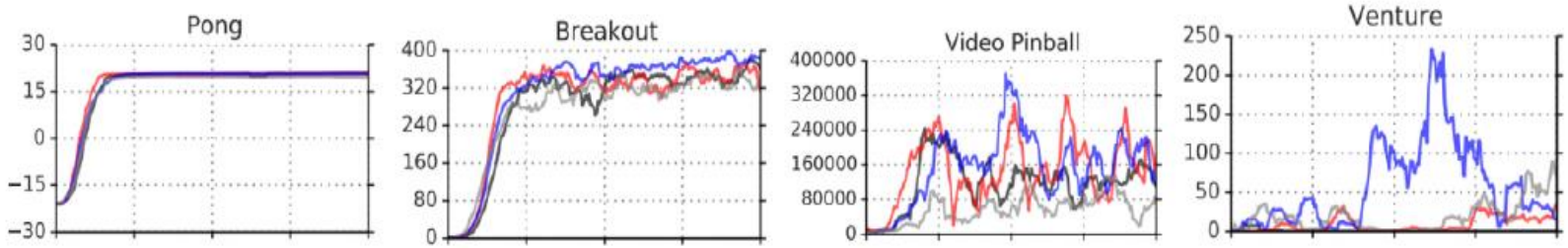
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a';\theta,\alpha) \right)$$

Dueling-DQN(ICML 2016)



训练技巧

1. 算法确定后，首先在简单可信的任务上训练(比如Atrai)，确保你的配置是正确的



2. 一般情况下，经验池越大，越有助于提升稳定性

3. 收敛过程曲折震荡，需要保持耐心(有时性能比随机动作还糟)

4. 初始epsilon值可以设置大一些，逐渐减小

总结

1. DQN(Deep Q learning)
   经验回放
   目标网络

2. Double DQN
   利用两个网络解决过估计问题

3. Prioritized Experience Replay
   利用TD误差对样本池数据进行筛选

4. Dueling DQN
   Q函数分解为值函数V和优势函数A