RL Tutorial 2

FIT 2017

DQN Tricks

- Double DQN
- max操作导致“过优化”
- Q^{target}(s, a)是目标，Q^{approx}是NN，Y是误差， a′=argmax_aQ^{target}(s′,a)

$$
\begin{aligned}
Z &= r_{s,a} + \gamma max_{a1} Q^{approx}(s', a1) - r_{s,a} + \gamma max_{a2} Q^{target}(s', a2) \\
&= \gamma max_{a1} Q^{approx}(s', a1) - \gamma max_{a2} Q^{target}(s', a2) \\
&\geq \gamma Q^{approx}(s', a') - Q^{target}(s', a') = \gamma Y_{s',a'}
\end{aligned}
$$

Double Q Network

- 于是DQN的估计不再是无偏的
- 解决办法：训两个Q网络，一个选择动作一个用于计算，交替更新

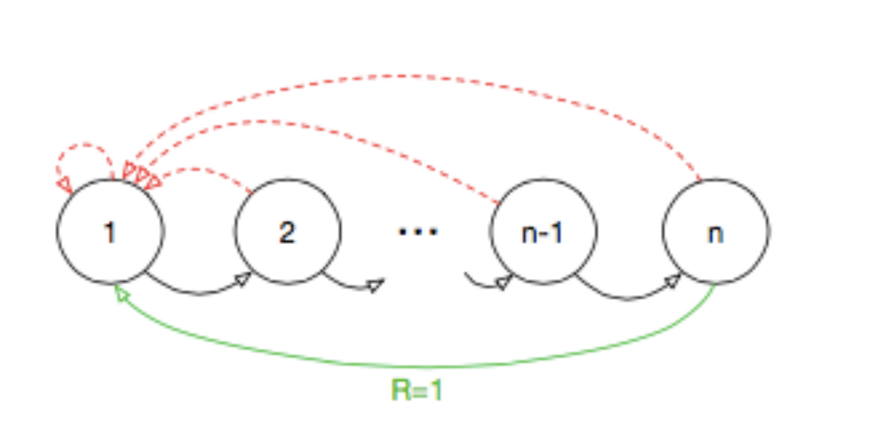
Algorithm 1 Double Q-learning

```
Initialize Q^A,Q^B,s
repeat
    Choose a, based on Q^A(s,·) and Q^B(s,·), 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) ← Q^A(s,a) + α(s,a)(r + γQ^B(s',a*) − Q^A(s,a))
    else if UPDATE(B) then
        Define b* = arg max_a Q^B(s',a)
        Q^B(s,a) ← Q^B(s,a) + α(s,a)(r + γQ^A(s',b*) − Q^B(s,a))
    end if
    s ← s'
until end
```

Prioritized Replay

- 使更Exciting的样本更容易被采样

- 对于奖励稀疏的问题效果很好，例如



- 维护优先队列

- 显然不能仅仅是贪婪的采样

Prioritized Replay

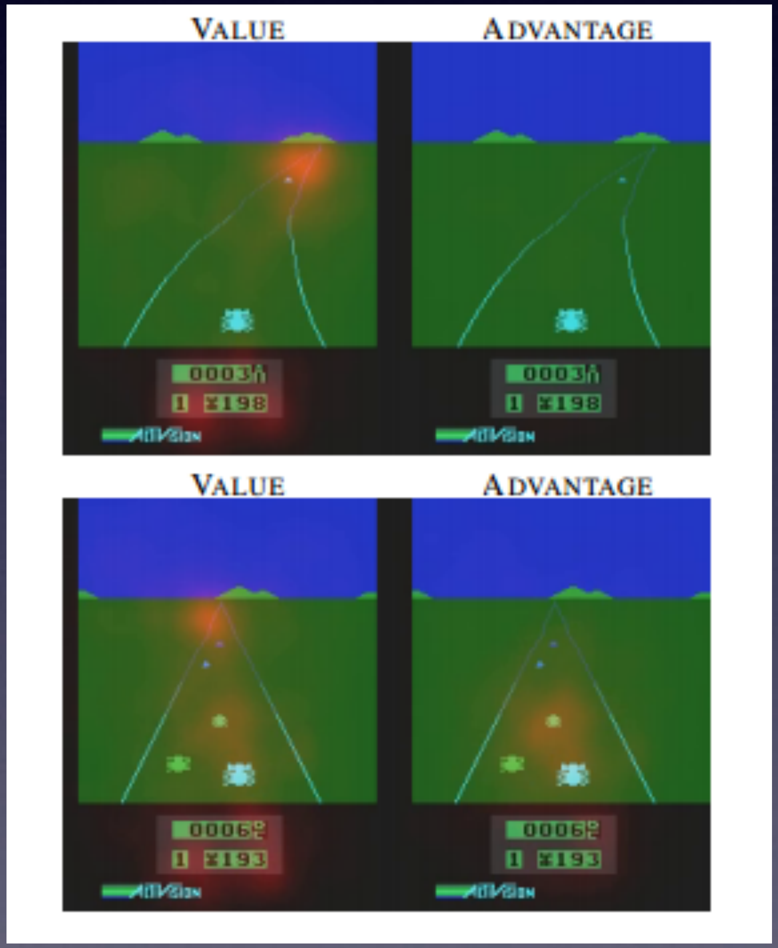
- 按照rank-based或者proportional
- 等于改变了样本分布，需要修偏

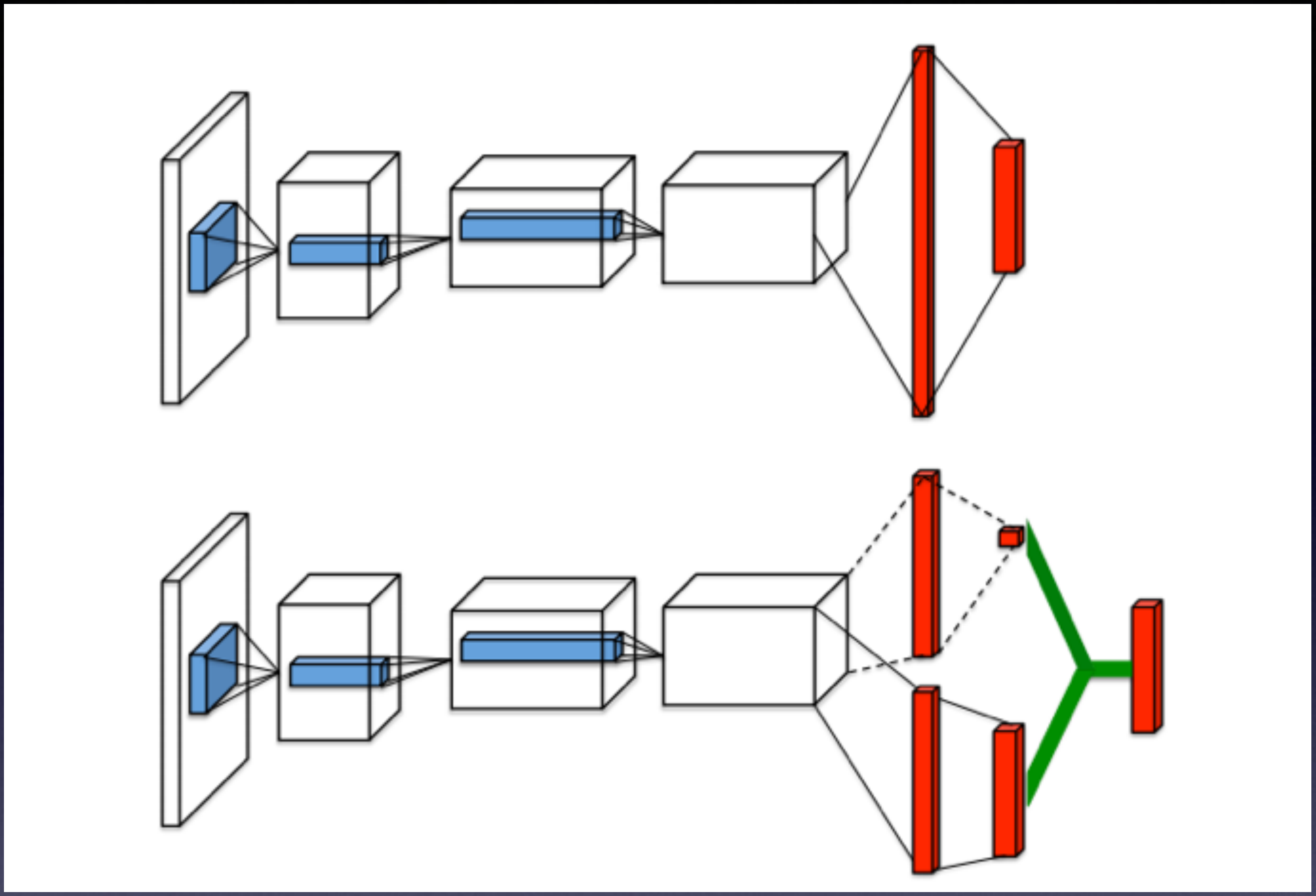
Algorithm 1 Double DQN with proportional prioritization

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

Dueling Network

- $Q(s, a) = V(s) + A(s, a)$
- 右图中前方没有车时
- Action随意





Dueling Network

基于价值的方法

- Q-Learning
- 估计状态-动作对的价值，选取argmax

基于策略的方法

- NN的输出直接就是动作

$$a = \pi(a|s,\theta)$$

基本思想

- 存在一个策略 $\pi(a|s;\theta)$
- 如果我们知道每个状态下正确的行动a*
- 转化为炼丹：$max_{\theta}\sum_{n=1}^{N} logp(a_n^*|s_n;\theta)$
- 然而我们并不知道a*

基本思想

- 和上次DQN时一样，我们可以收集agent和环境互动的数据，从而得到一个状态-行动-收益序列

$$\tau = ((s_0, a_0, r_0), ..., (s_{T-1}, a_{T-1}, r_{T-1}), s_T)$$

- 例如，我们令R是序列收益之和
- 那么期望

$$\hat{E} = R \cdot \prod_{t=0}^{T-1} \pi(a_t | s_t; \theta)$$

基本思想

$$\hat{E} = R \cdot \prod_{t=0}^{T-1} \pi(a_t|s_t;\theta)$$

$$\hat{g} = \hat{E}\text{'} = R \cdot \nabla_\theta(\prod_{t=0}^{T-1} \pi(a_t|s_t;\theta))$$

- 通常使用log likelihood

$$R \cdot \nabla_\theta(\sum_{t=0}^{T-1} log\pi(a_t|s_t;\theta)))$$

- 使用梯度下降来优化刚才的期望

策略梯度

- 即策略期望总收益的梯度，一般记作 $\nabla_\theta J(\theta)$
- 可以证明 $E[\hat{g}]$ 是对它的无偏估计
- 换言之，确定型策略梯度算法给出的期望恰好就是策略梯度 (DPG原始论文证明)
- 优势：处理连续场景

DPG

Theorem 1 (Deterministic Policy Gradient Theorem). *Suppose that the MDP satisfies conditions A.1 (see Appendix; these imply that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s,a)$ exist and that the deterministic policy gradient exists. Then,*

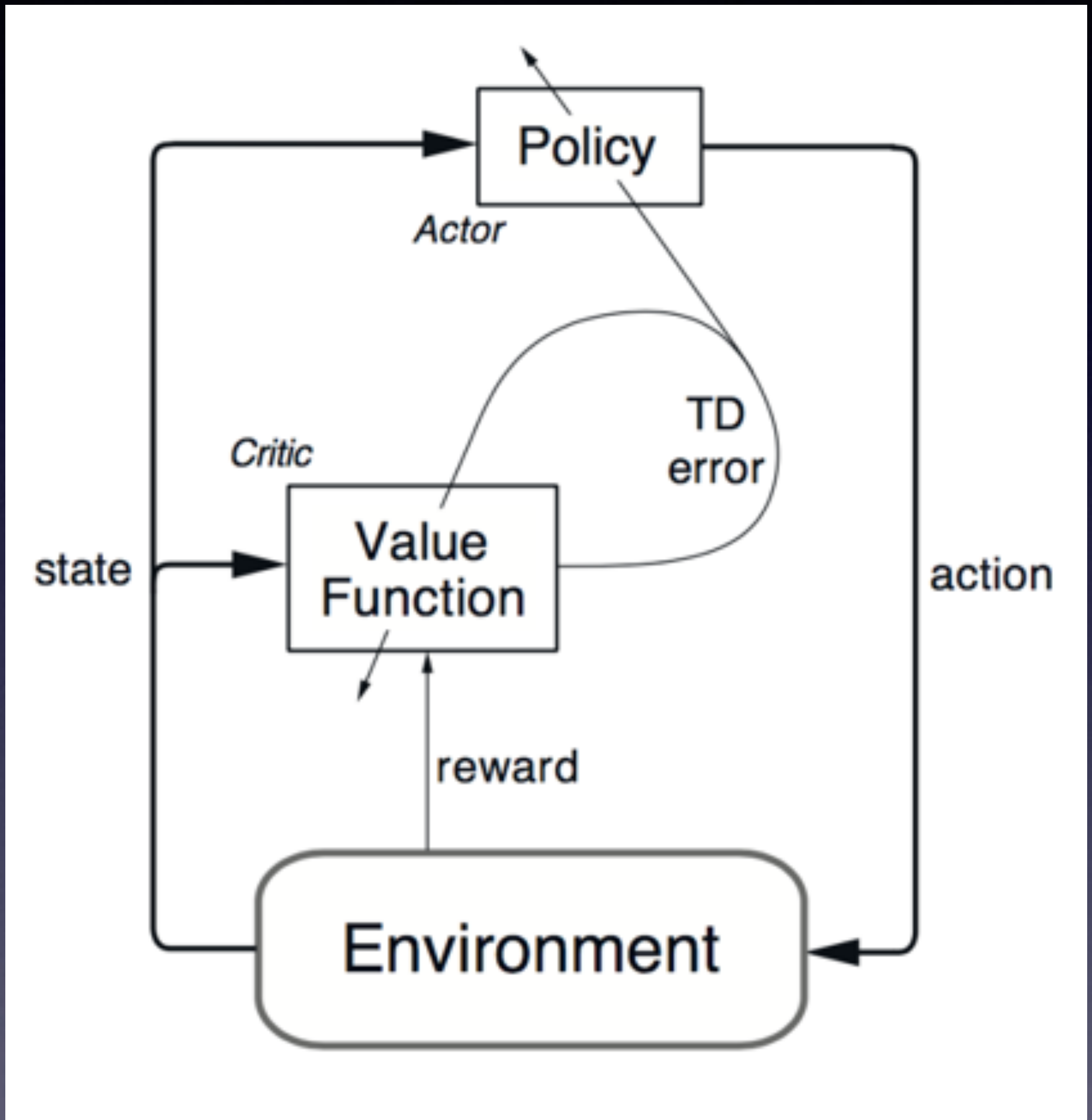
$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \, \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} \, \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^\mu} \left[\nabla_\theta \mu_\theta(s) \, \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} \right] \quad (9) \end{aligned}$$

- 确定性策略梯度
- 按照Q最大的方向调整策略梯度

Actor-Critic

- 求解策略梯度
- 两个网络
- Actor是策略，参数为u，$\pi(s; u)$ 输出action
- Critic是值函数，参数为w，$Q(s, a; w)$ 输出Q值
- Actor为Critic决定a
- Critic为Actor提供Loss function，评判结果好坏

- Critic: Q-Learning Methods
- Actor: 用Q值计算梯度更新策略



- We use a critic to estimate the action-value function,

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$

- Actor-critic algorithms maintain *two* sets of parameters
  - Critic: Updates action-value function parameters w
  - Actor: Updates policy parameters θ, in direction suggested by critic
- Actor-critic algorithms follow an *approximate* policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$$

而不再是Q π (s, a)

- Using linear value fn approx. $Q_w(s,a) = \phi(s,a)^\top w$
  - Critic: Updates w by linear TD(0)
  - Actor: Updates θ by policy gradient

function QAC
  Initialise s, θ
  Sample $a \sim \pi_\theta$
  for each step **do**
    Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
    Sample action $a' \sim \pi_\theta(s', a')$
    $\delta = r + \gamma Q_w(s',a') - Q_w(s,a)$
    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)$
    $w \leftarrow w + \beta \delta \phi(s,a)$
    $a \leftarrow a', s \leftarrow s'$
  end for
end function

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**
    noise sample
    Select action $\boxed{a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t}$ 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
    experience replay
    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 Critic Network
    Update the actor policy using the sampled gradient:

update Actor Network

Critic target Network

Actor target Network

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

    Update the target networks:

$$\boxed{\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \end{aligned}}$$

update Actor & Critic Target Network

  end for
end for

DDPG

Blue tricks

- The policy gradient has many equivalent forms

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta\right] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta e\right] && \text{TD}(\lambda)\text{ Actor-Critic} \\
G_\theta^{-1}\nabla_\theta J(\theta) &= w && \text{Natural Actor-Critic}
\end{aligned}
$$

- Each leads a stochastic gradient ascent algorithm
- Critic uses policy evaluation (e.g. MC or TD learning) to estimate $Q^\pi(s,a)$, $A^\pi(s,a)$ or $V^\pi(s)$

Policy Gradient的变种

A3C

- Asynchronous Advantage Actor-Critic
- 异步：用多线程替代Experience Replay
- 梯度 $\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t, a_t; \theta, \theta_v)$
- 选择Advantage最大而非Q最大
- A是n-step TD Error $\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v)$

- Estimate state-value function

$$V(s, \mathbf{v}) \approx \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \ldots | s\right]$$

- Q-value estimated by an n-step sample

$$q_t = r_{t+1} + \gamma r_{t+2} \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}, \mathbf{v})$$

- Actor is updated towards target

$$\frac{\partial l_u}{\partial \mathbf{u}} = \frac{\partial \log \pi(a_t | s_t, \mathbf{u})}{\partial \mathbf{u}} (q_t - V(s_t, \mathbf{v}))$$

- Critic is updated to minimise MSE w.r.t. target

$$l_v = (q_t - V(s_t, \mathbf{v}))^2$$

A3C流程

Algorithm S2 Asynchronous n-step Q-learning - pseudocode for each actor-learner thread.

// Assume global shared parameter vector θ.
// Assume global shared target parameter vector θ^-.
// Assume global shared counter $T = 0$.
Initialize thread step counter $t \leftarrow 1$
Initialize target network parameters $\theta^- \leftarrow \theta$
Initialize thread-specific parameters $\theta' = \theta$
Initialize network gradients $d\theta \leftarrow 0$
repeat
  Clear gradients $d\theta \leftarrow 0$
  Synchronize thread-specific parameters $\theta' = \theta$
  $t_{start} = t$
  Get state s_t
  repeat
    Take action a_t according to the ϵ-greedy policy based on $Q(s_t, a; \theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$

训练数据(s,a,r)序列生成，
最大t_max个

  $R = \begin{cases} 0 & \text{for terminal } s_t \\ \max_a Q(s_t, a; \theta^-) & \text{for non-terminal } s_t \end{cases}$

n-step TD error

  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \frac{\partial \left(R - Q(s_i, a_i; \theta')\right)^2}{\partial \theta'}$ 累积"batch"内的梯度
  end for
  Perform asynchronous update of θ using $d\theta$. 异步更新网络参数
  if $T \mod I_{target} == 0$ **then**
    $\theta^- \leftarrow \theta$ 低频率同步target网络参数
  end if
until $T > T_{max}$

具体算法

Examples

- https://github.com/dennybritz/reinforcement-learning 各类资源、习题与答案

- https://github.com/yanpanlau/DDPG-Keras-Torcs 300行DDPG训练TORCS自动驾驶（Keras+TF）

- https://github.com/NVlabs/GA3C 核弹厂的A3C

Thanks!

Q&A?