Reinforcement Learning workshop

Florian Goebels Sep 2019

September 19th 2019

aggregate intellect

Agenda

- Day 1: RL Basics (theory)
- Day 2: Q-learning (coding)
- Day 3: Policy Gradient methods (coding)

Today's Agenda

- Day 2: Q-Learning
  - Q-Learning
  - Deep Q-Learning
  - Rainbow

Q-Learning

- Combines ideas from DP and MC methods
    - Like MC: learns directly from raw experiences
    - Like DP: iteratively updates estimates

TD *V(S)* estimate

MC

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t) \right]$$

DP

$$v_\pi(s) = \mathbf{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s \right]$$

TD

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1} - V(S_t)) \right]$$

TD *V(S)* estimate

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1} - V(S_t)\right]$$

?

Better estimate of *V(S)*

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

TD ERROR

Current estimate of *V(S)*

TD advantages

- Does not require model of environment (unlike DP)
- MC needs to wait until episode finish, TP can online update
    - MC it’s hard to estimate value of action-state pair
- Both TD and MC converge to optimal policy?

TD demo

- https://cs.stanford.edu/people/karpathy/reinforcejs/

Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Deep Q-Learning

Deep Q-Learning

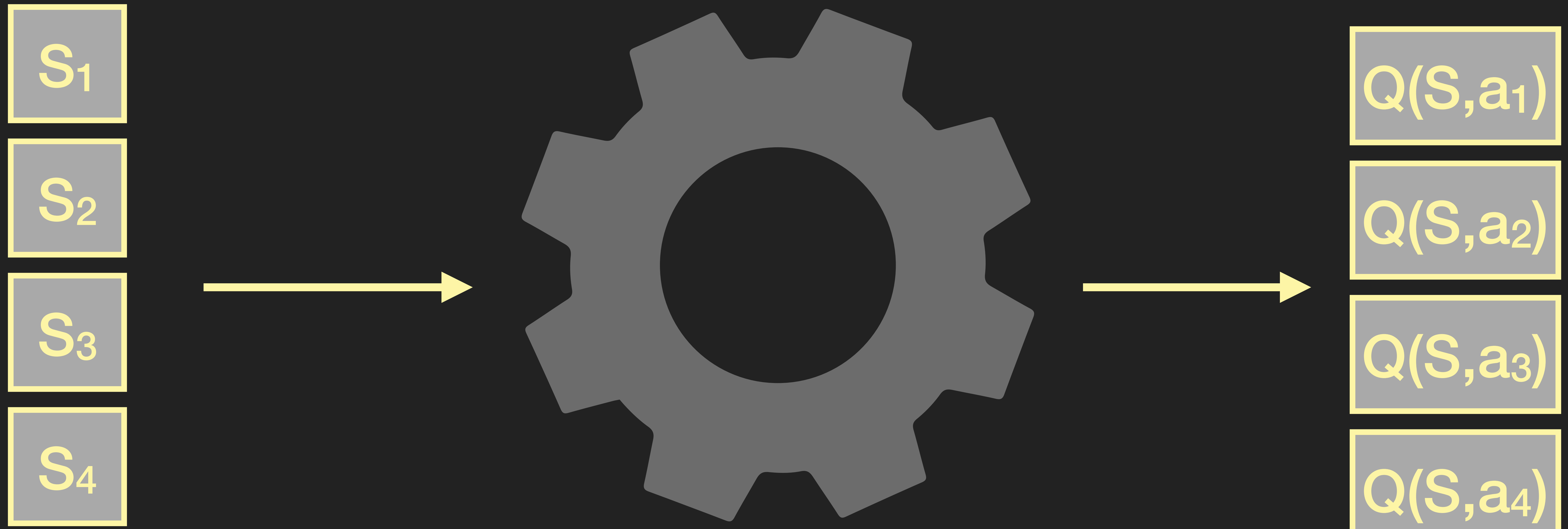
- In Q learning we iteratively update ***Q(A,S)*** via policy iteration
- Neural networks are general function approximators NN(S) -> V(A)
- Cost function:

Better estimate of *V(S)*

$$\left(\mathbf{NN}(S_t, a) - r + \gamma \max_a \mathbf{NN}(S_{t+1}, a)\right)^2$$

NN estimation of *V(S)*

Deep Q-Learning in a nutshell



Deep Q-Learning

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

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 $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

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

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

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

End For

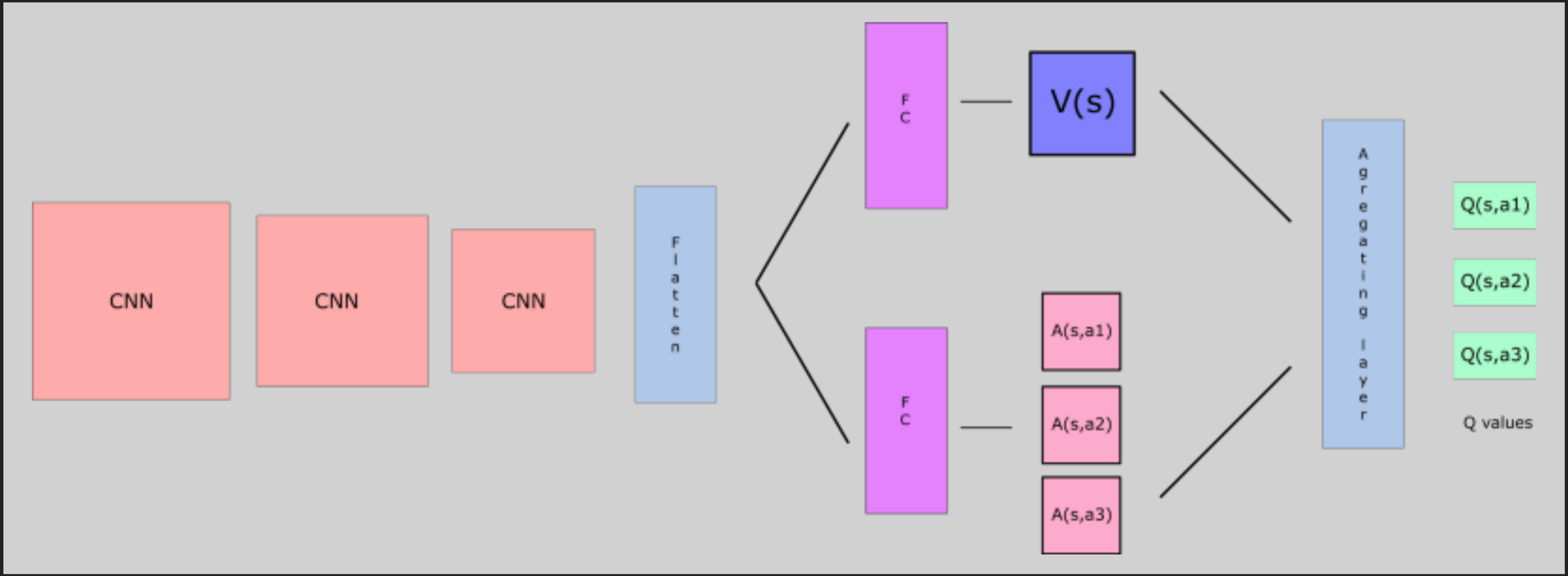
End For

Double deep Q-Learning

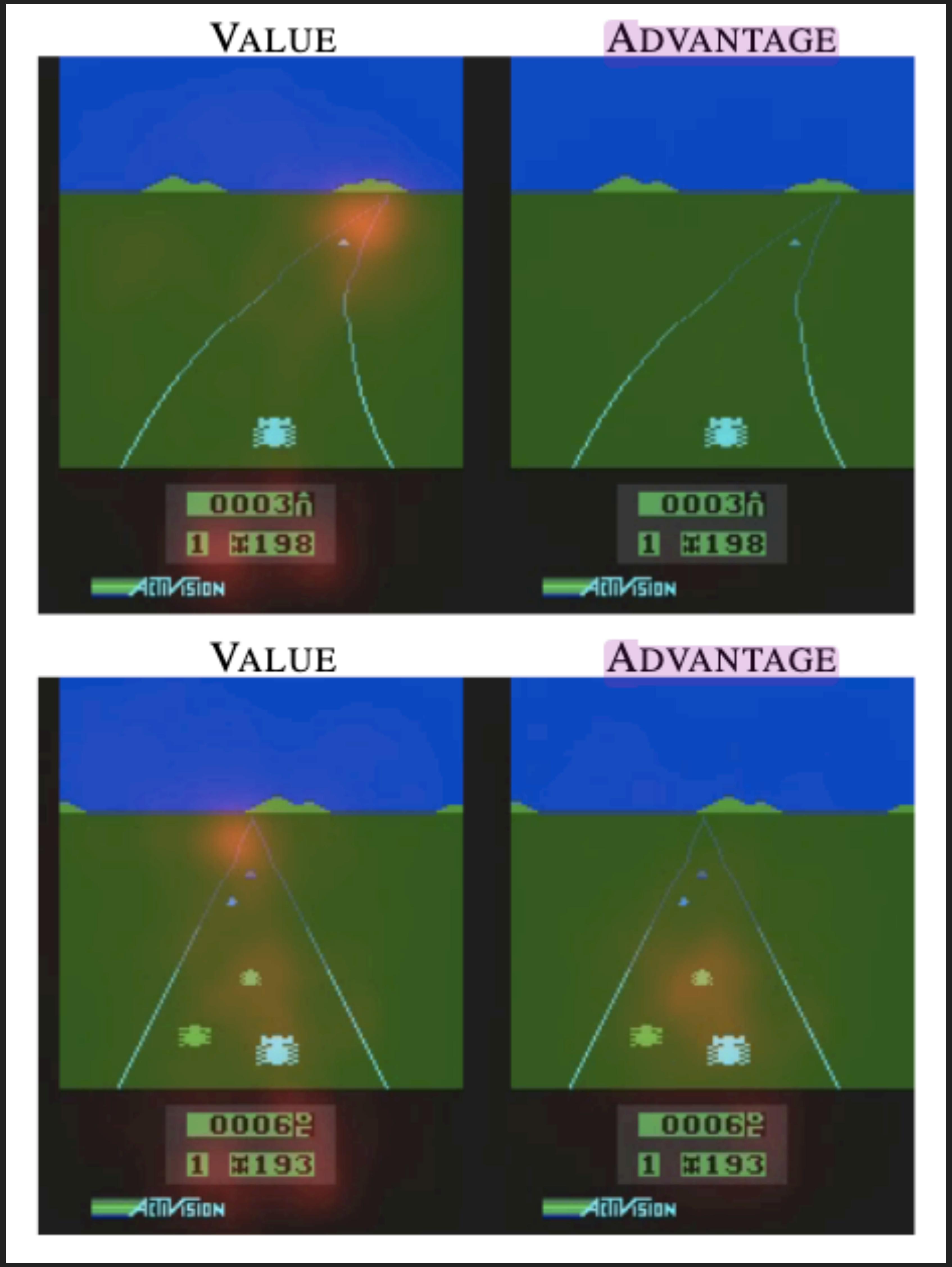
$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

$$Y_t^{\mathrm{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t')$$

Duelling deep Q-Learning

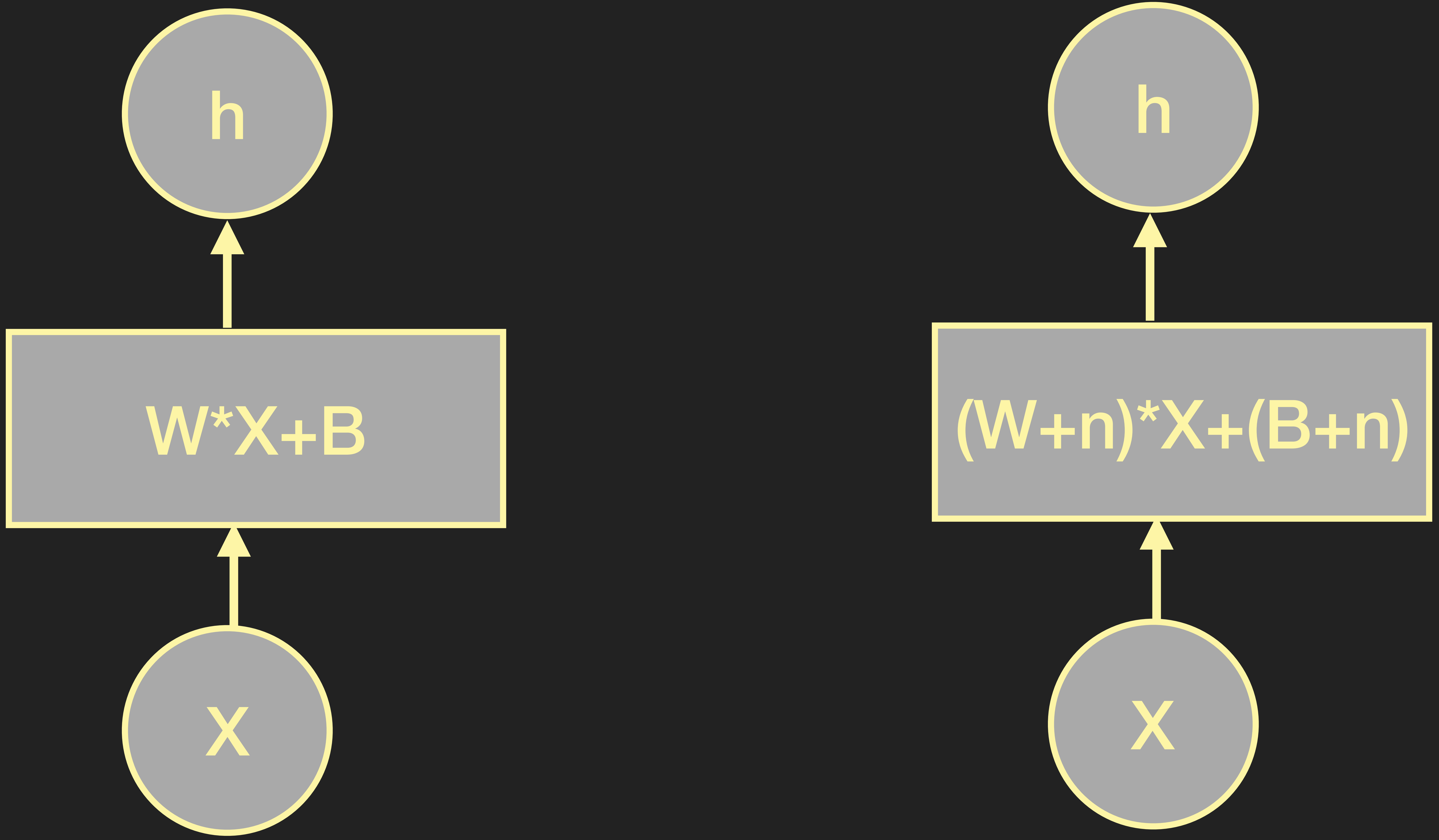


Advantage of Action



- Value stream pays attention to road
- Advantage stream pays to care in front to avoid collision
- Final output V(s) + (A - mean(A))
- How much better is action than other in average

Noisy Nets



Multi-step learning

$$\left(\mathbf{NN}(S_t, a) - r + \gamma \max_a \mathbf{NN}(S_{t+1}, a)\right)^2$$

$$\left(\mathbf{NN}(S_t, a) - r + \gamma^n \max_a \mathbf{NN}(S_{t+n}, a)\right)^2$$

Prioritized Experience Replay

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

Distributional Reinforcement Learning

License