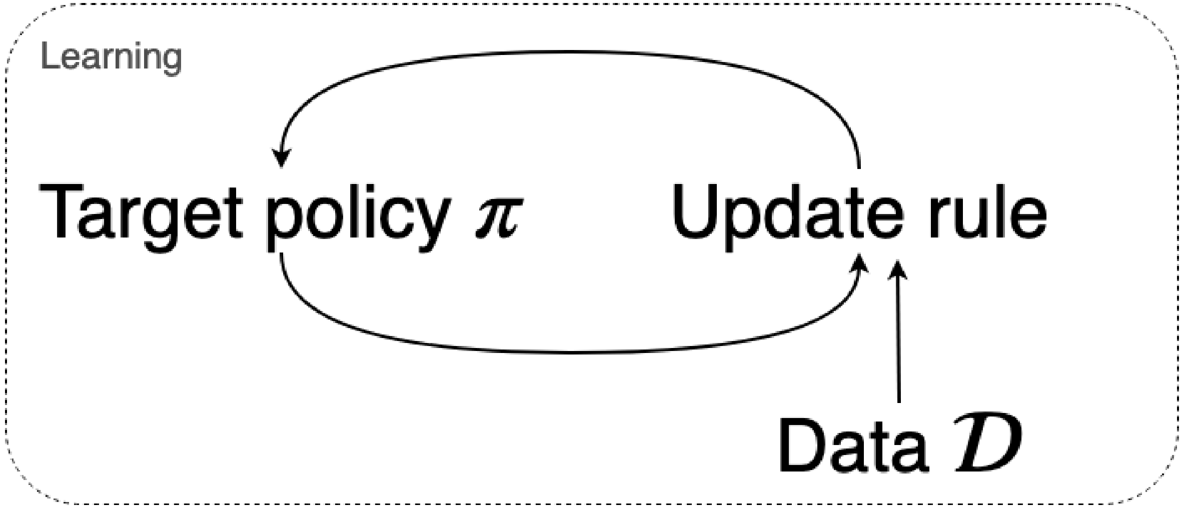
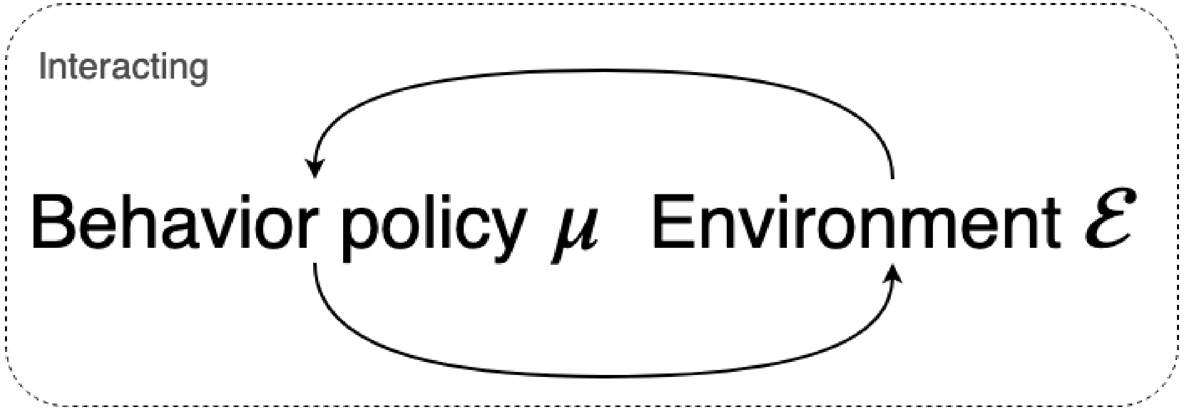
ETH zürich

Off-policy Learning and the Deadly Triad

Deep Reinforcement Learning Seminar

Alexander Nedergaard





Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

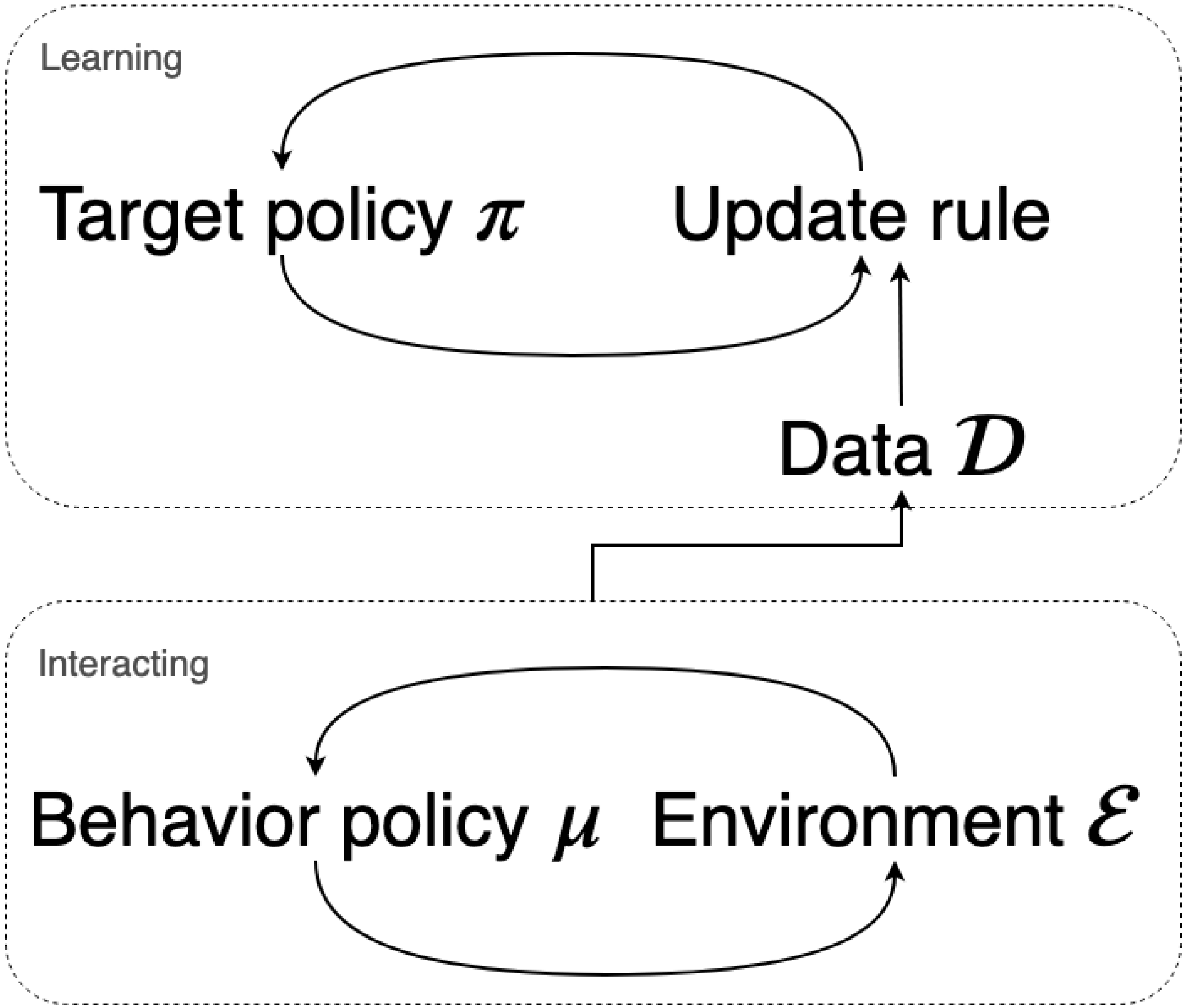
Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \underset{\tau \sim \pi_k}{\mathrm{E}}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for



Algorithm 5 PPO with Clipped Objective

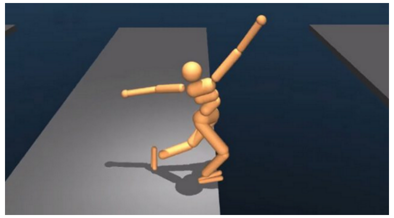
Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

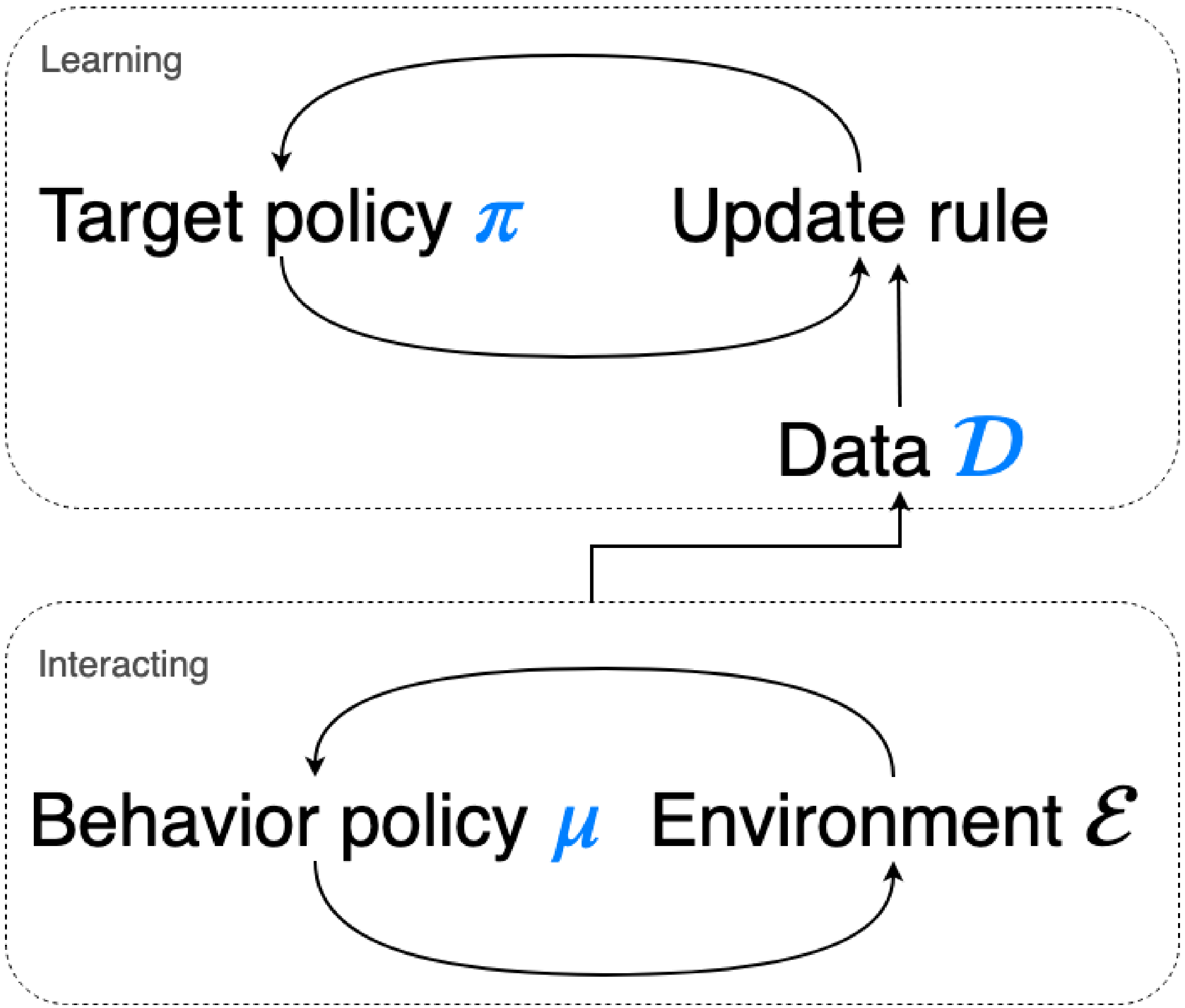
  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for



$$\pi = \mu$$



Algorithm 5 PPO with Clipped Objective

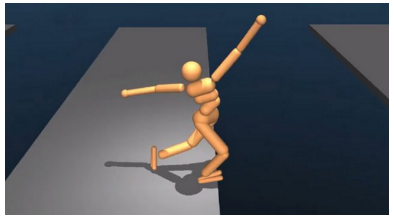
Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

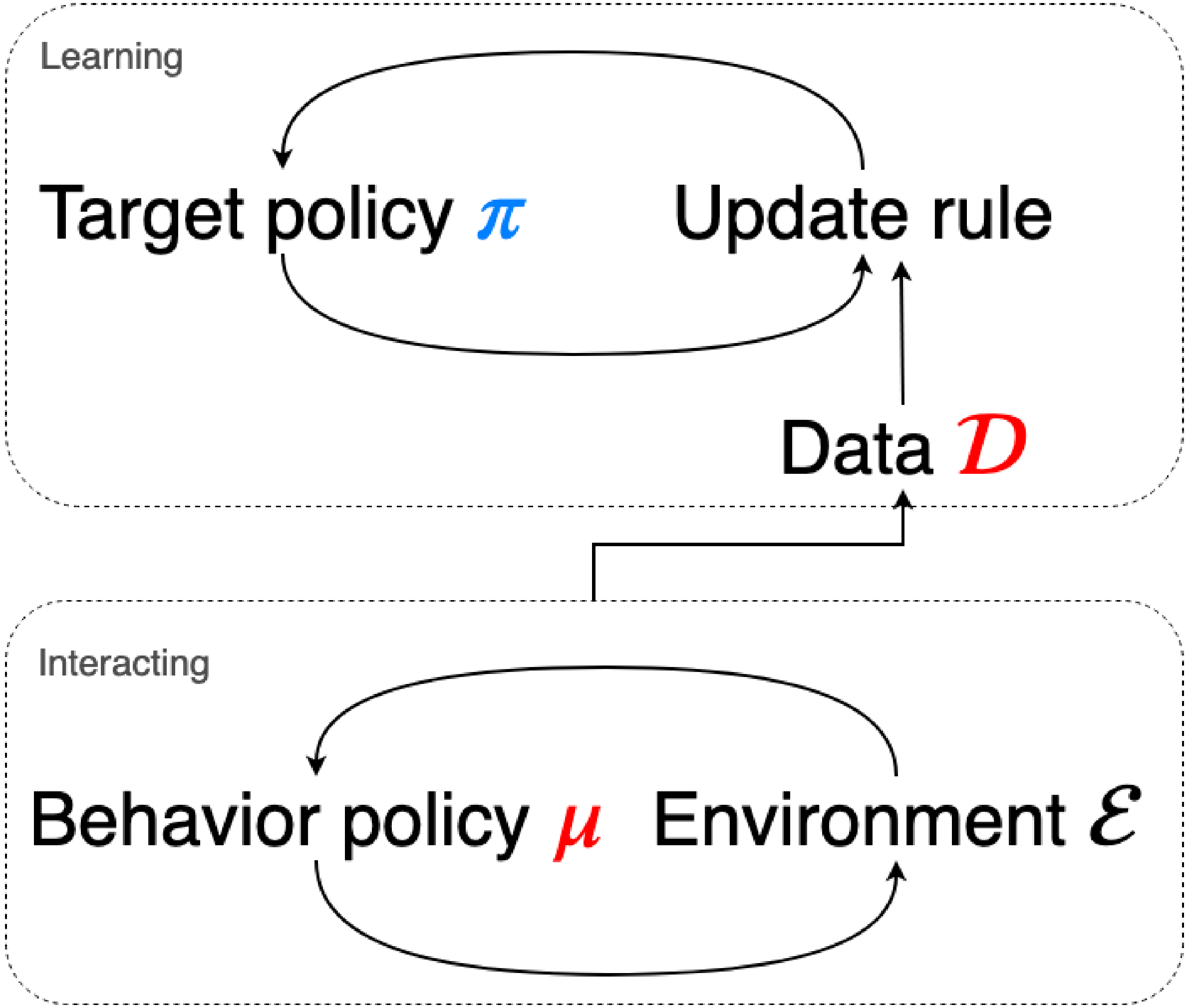
  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for



$\pi \neq \mu$



Algorithm 5 PPO with Clipped Objective

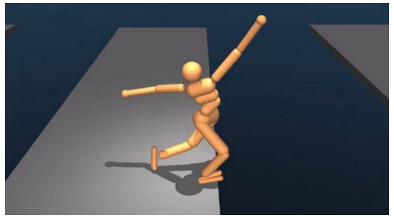
Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

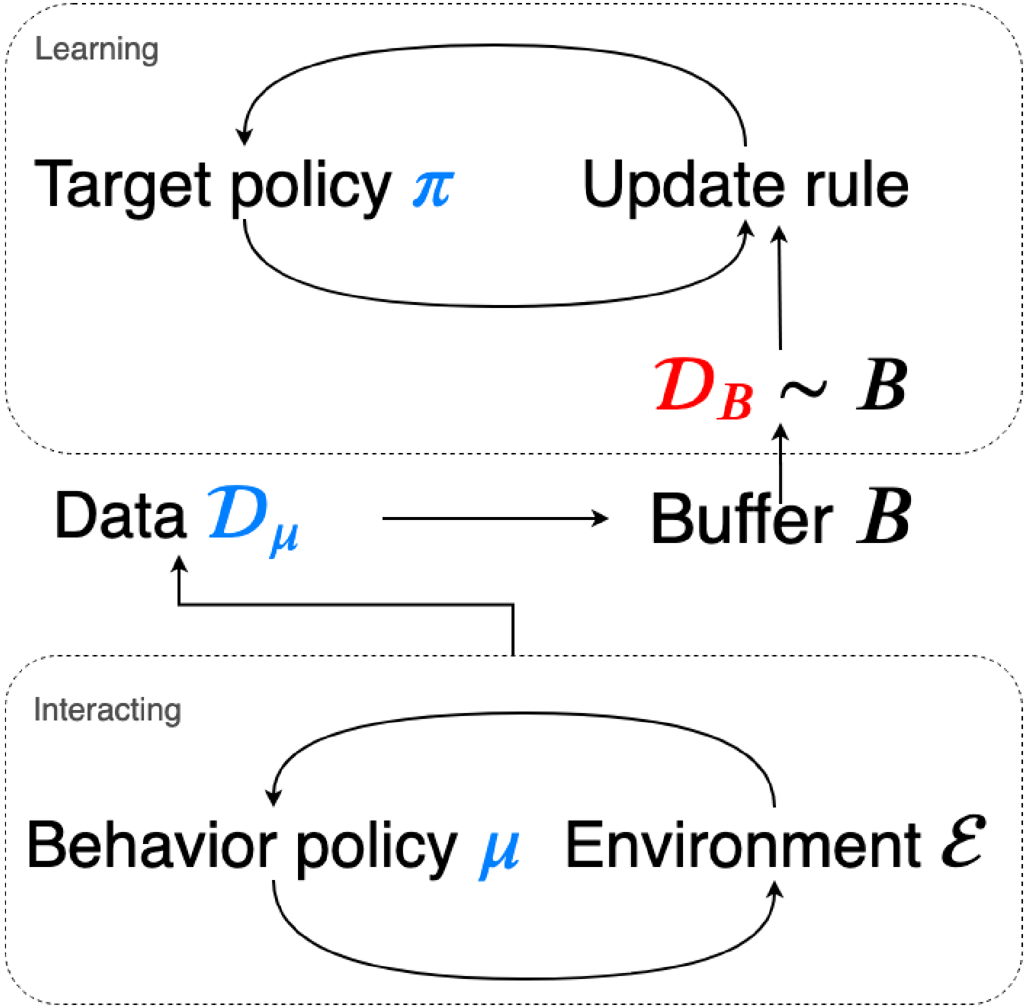
by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

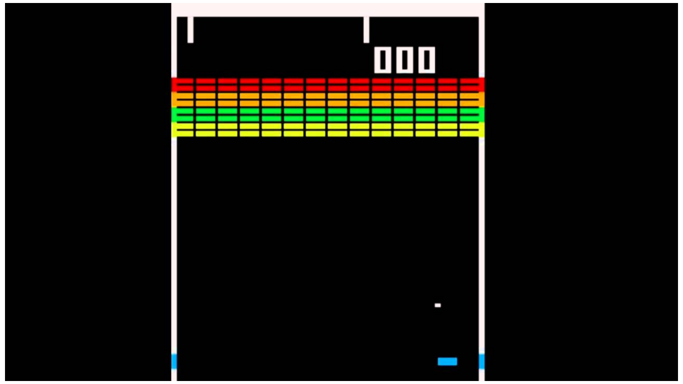


Experience Replay (Lin, 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching)

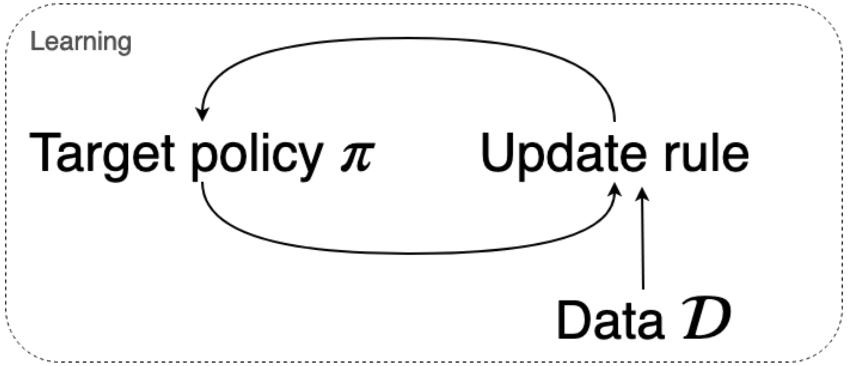


Algorithm 1 Deep Q-learning with Experience Replay

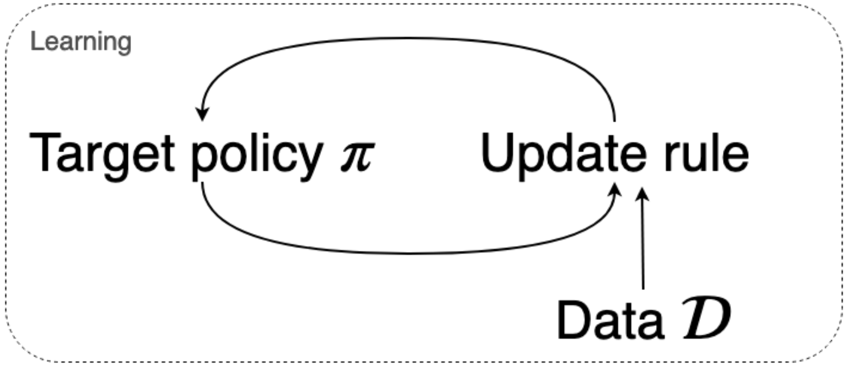
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise 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



SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)

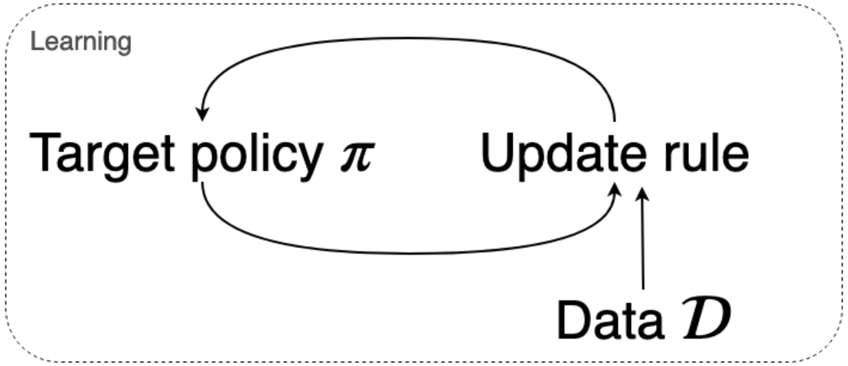


SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)



$$\pi(s) = \arg\max_a Q(s, a)$$

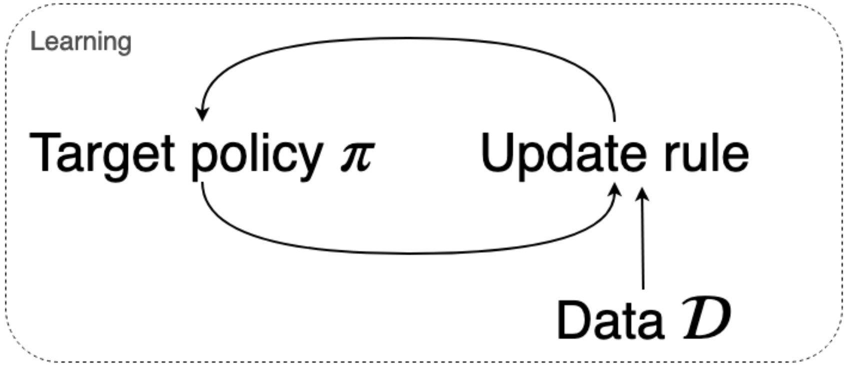
SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)



$$\pi(s) = \arg\max_a Q(s,a)$$

$$\text{Update rule: } Q(s,a) \leftarrow r + \gamma Q(s',a')$$

SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)

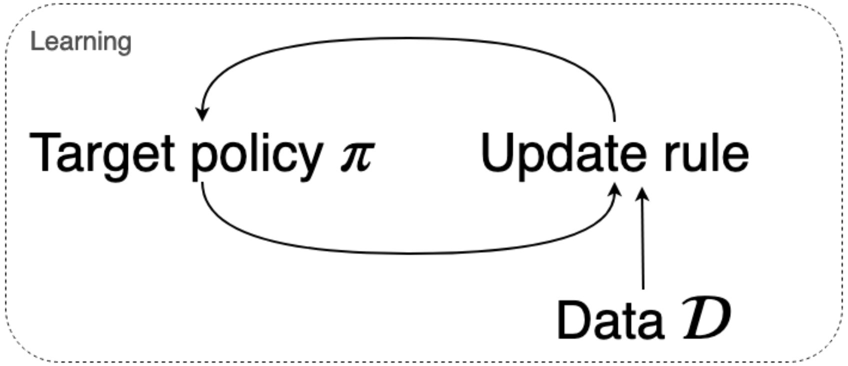


$$\pi(s) = \arg\max_a Q(s,a)$$

$$\text{Update rule: } Q(s,a) \leftarrow r + \gamma Q(s', a')$$

$$\mathcal{D} : \{(s_i, a_i, r_i, s'_i, a'_i)\}_i$$

SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)



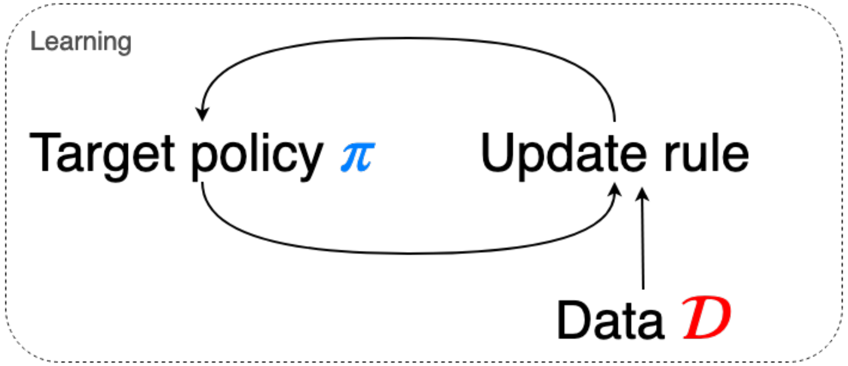
$$\pi(s) = \arg\max_a Q(s,a)$$

$$\text{Update rule: } Q(s,a) \leftarrow r + \gamma Q(s', a')$$

$$\mathcal{D} : \{(s_i, a_i, r_i, s'_i, a'_i)\}_i$$

$$\text{Bellman equation: } Q^\pi(s,a) = r + \gamma \mathbb{E}_{s' \sim \mathcal{E}}[\mathbb{E}_{a' \sim \pi}[Q^\pi(s', a')]]$$

SARSA (Rummery and Niranjan, 1994. On-line Q-learning using connectionist systems)



$$\pi(s) = \arg\max_a Q(s,a)$$

$$\text{Update rule: } Q(s,a) \leftarrow r + \gamma Q(s', a')$$

$$\mathcal{D} : \{(s_i, a_i, r_i, s'_i, a'_i)\}_i$$

$$\text{Bellman equation: } Q^\pi(s,a) = r + \gamma \mathbb{E}_{s' \sim \mathcal{E}}[\mathbb{E}_{a' \sim \pi}[Q^\pi(s', a')]]$$

SARSA

Update rule: $Q(s,a) \leftarrow r + \gamma Q(s', a')$

$\mathcal{D} : \{(s_i, a_i, r_i, s'_i, a'_i)\}_i$

Q-learning (Watkins and Dayan, 1992)

Update rule: $Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s', a')$

$\mathcal{D} : \{(s_i, a_i, r_i, s'_i)\}_i$

Bellman equation: $Q^\pi(s,a) = r + \gamma \mathbb{E}_{s' \sim \mathcal{E}}[\mathbb{E}_{a' \sim \pi}[Q^\pi(s', a')]]$

SARSA

Update rule: $Q(s, a) \leftarrow r + \gamma Q(s', a')$

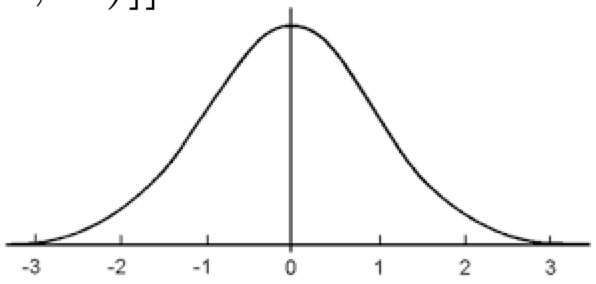
$\mathcal{D} : \{(s_i, a_i, r_i, s'_i, a'_i)\}_i$

Q-learning (Watkins and Dayan, 1992)

Update rule: $Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$

$\mathcal{D} : \{(s_i, a_i, r_i, s'_i)\}_i$

Bellman equation: $Q^\pi(s, a) = r + \gamma \mathbb{E}_{s' \sim \mathcal{E}}[\mathbb{E}_{a' \sim \pi}[Q^\pi(s', a')]]$

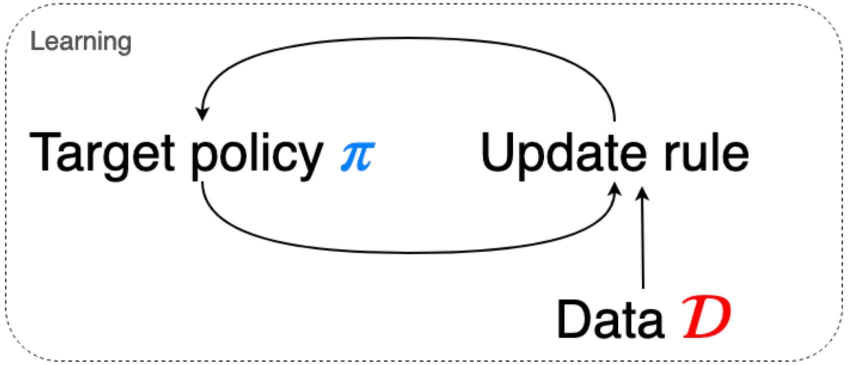


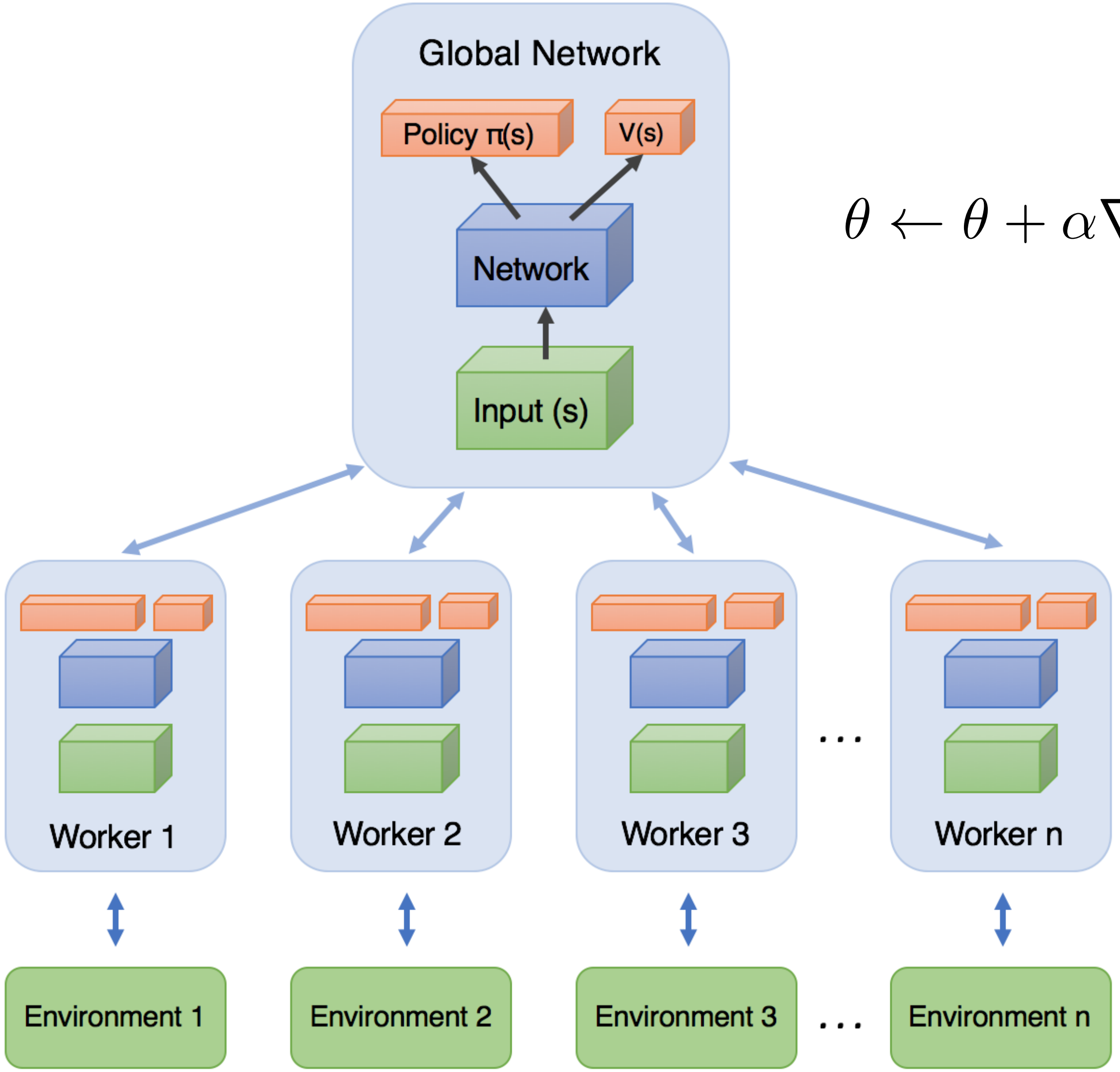
A3C (Mnih et al., 2016. Asynchronous Methods for Deep Reinforcement Learning)

$$\text{Update rule:} \quad \theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta)$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_t (V^{\pi_\theta}(s_t) - b) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

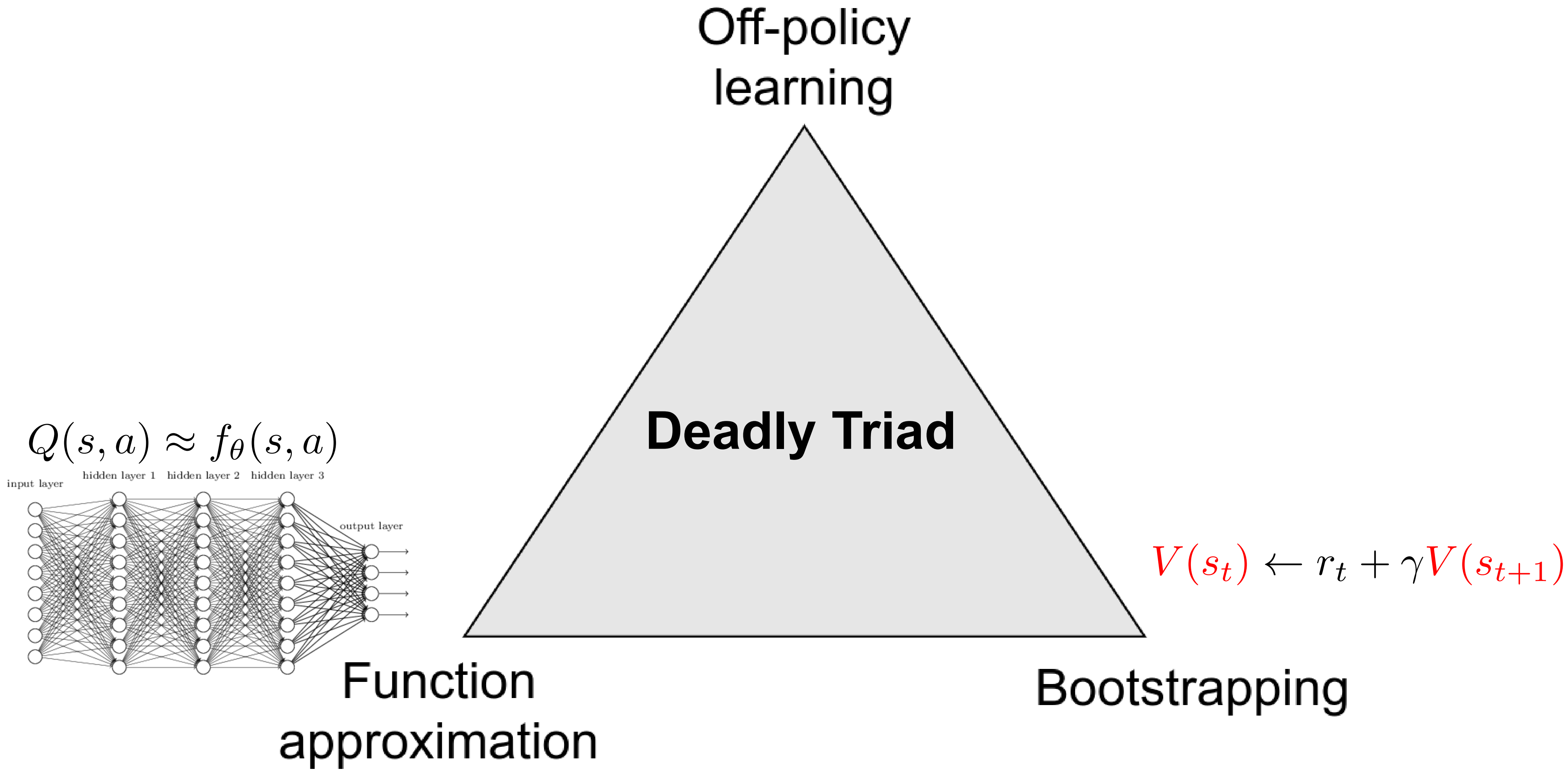
$$\mathcal{D} : \{\tau_i\}_i$$





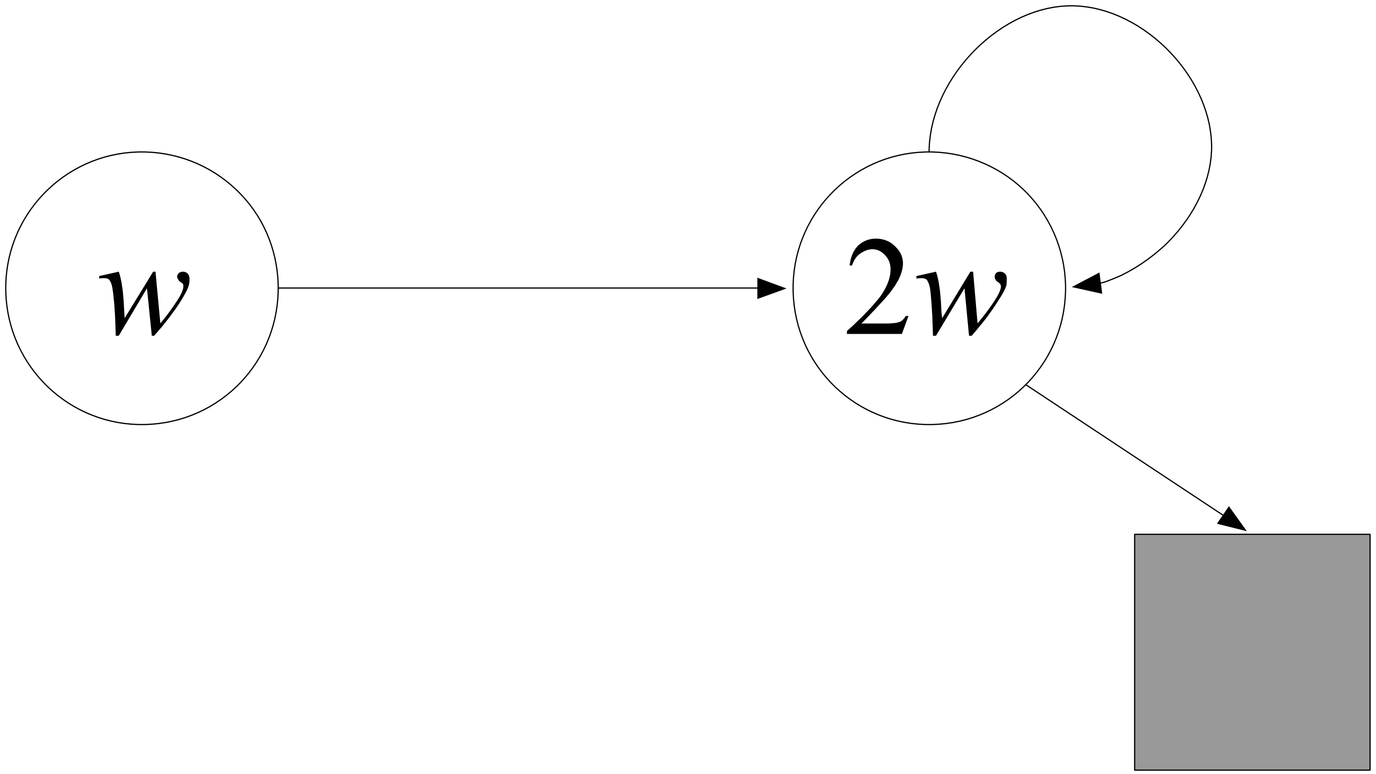
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta)$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_t (V^{\pi_\theta}(s_t) - b) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

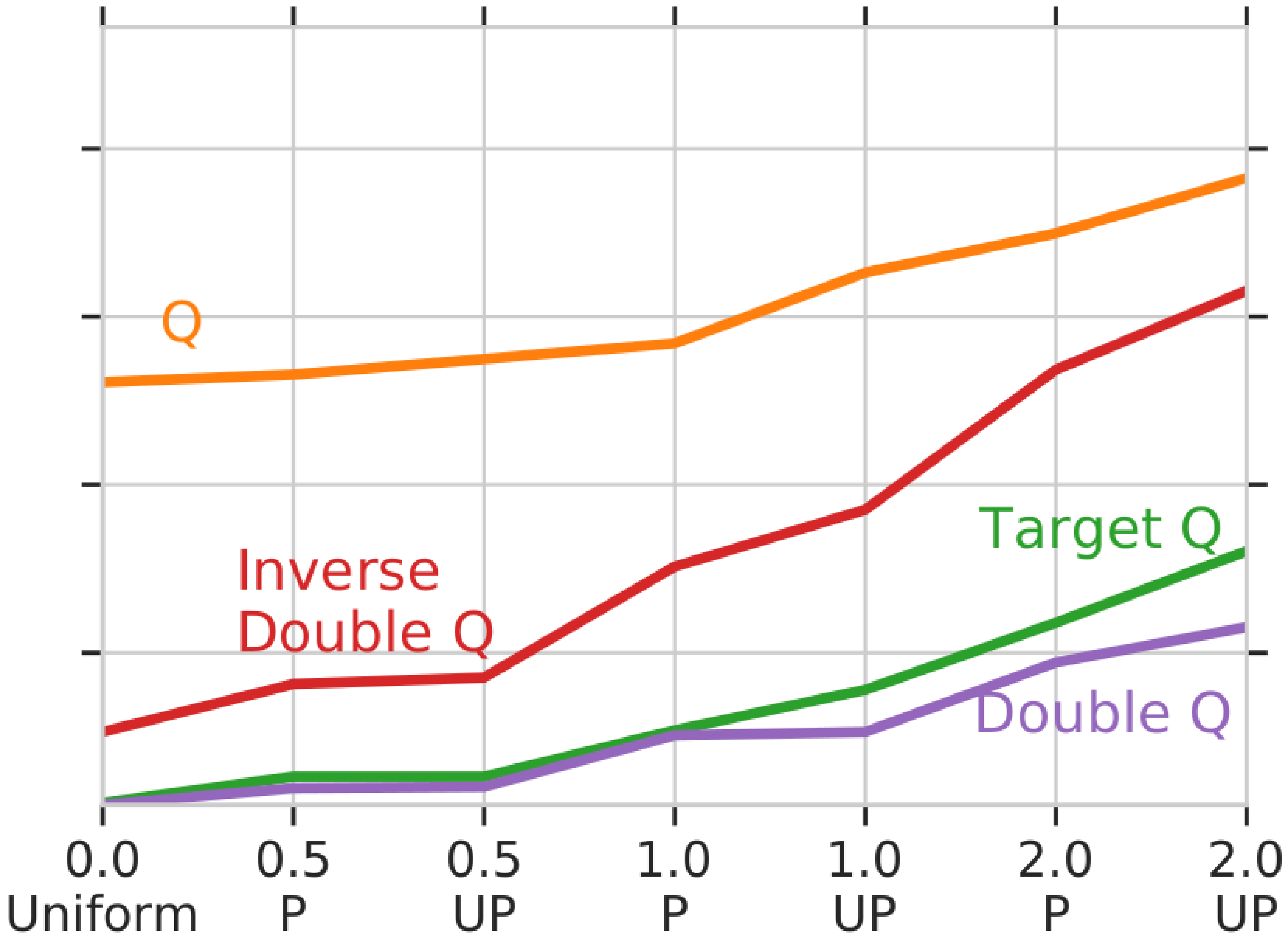


Off-policy learning
Deadly Triad
$Q(s,a) \approx f_\theta(s,a)$
input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer
Function approximation
$V(s_t) \leftarrow r_t + \gamma V(s_{t+1})$
Bootstrapping

TD convergence with linear function approximation (Tsitsiklis and Van Roy, 1997. An Analysis of Temporal-Difference Learning with Function Approximation)

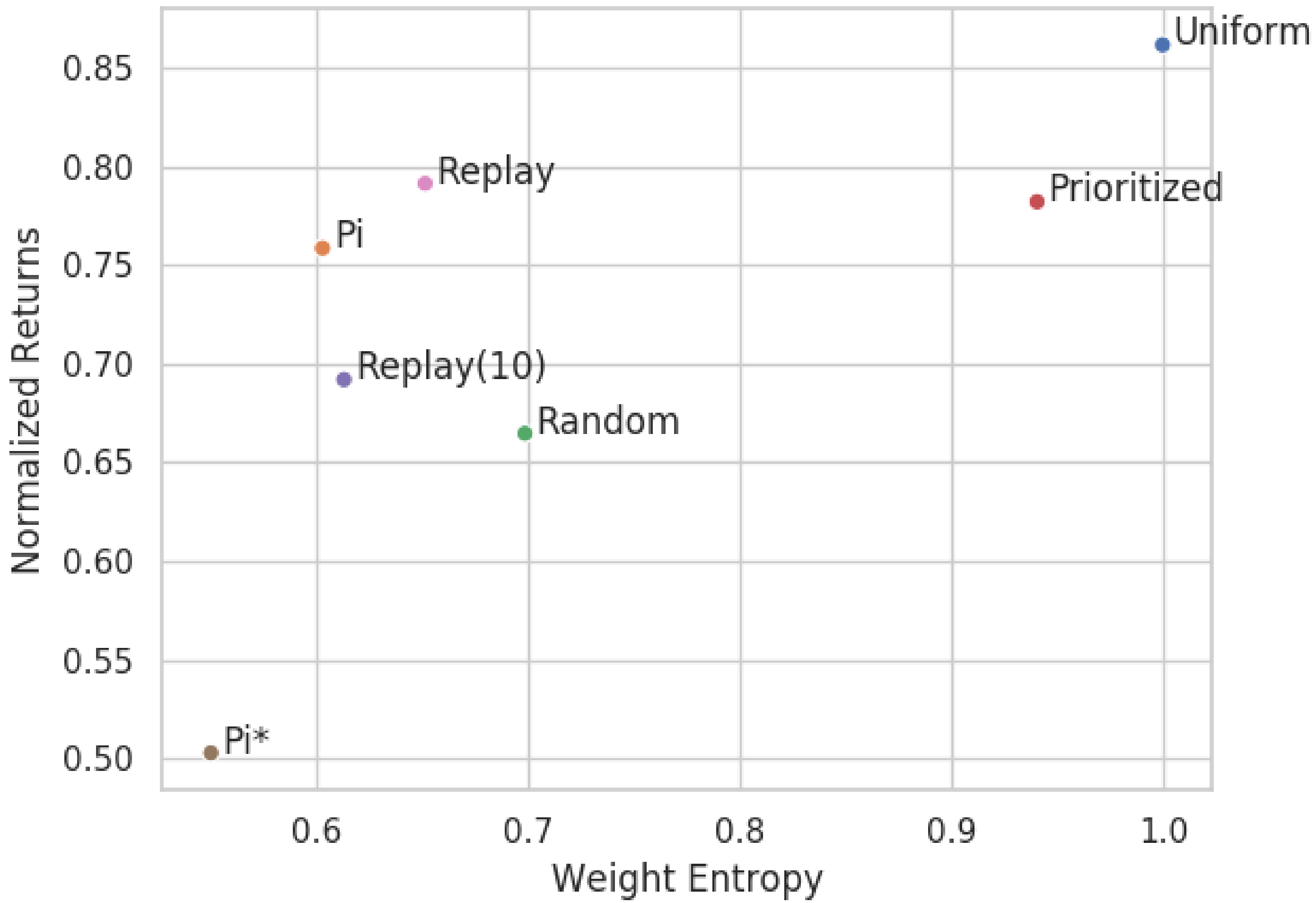


Less uniform sampling in Prioritized Experience Replay increases divergence in DQN (Van Hasselt et al., 2018. Deep Reinforcement Learning and the Deadly Triad)



(c) Prioritization

Improved performance with more uniform sampling distributions in Fitted Q Learning (Fu et al., 2019. Diagnosing Bottlenecks in Deep Q-learning Algorithms)



Understanding contribution of off-policy learning to divergence in DQN using Neural Tangent Kernel (Achiam et al., 2019. Towards Characterizing Divergence in Deep Q learning)

$$Q_\theta \leftarrow Q_\theta + \alpha K_\theta {\color{red}D_p}(\mathcal{T}^* Q_\theta - Q_\theta) + \mathcal{O}(||\alpha g||^2)$$

Understanding contribution of off-policy learning to divergence in DQN using Neural Tangent Kernel (Achiam et al., 2019. Towards Characterizing Divergence in Deep Q learning)

$$Q_\theta \leftarrow Q_\theta + \alpha K_\theta D_p(\mathcal{T}^* Q_\theta - Q_\theta) + \mathcal{O}(||\alpha g||^2)$$

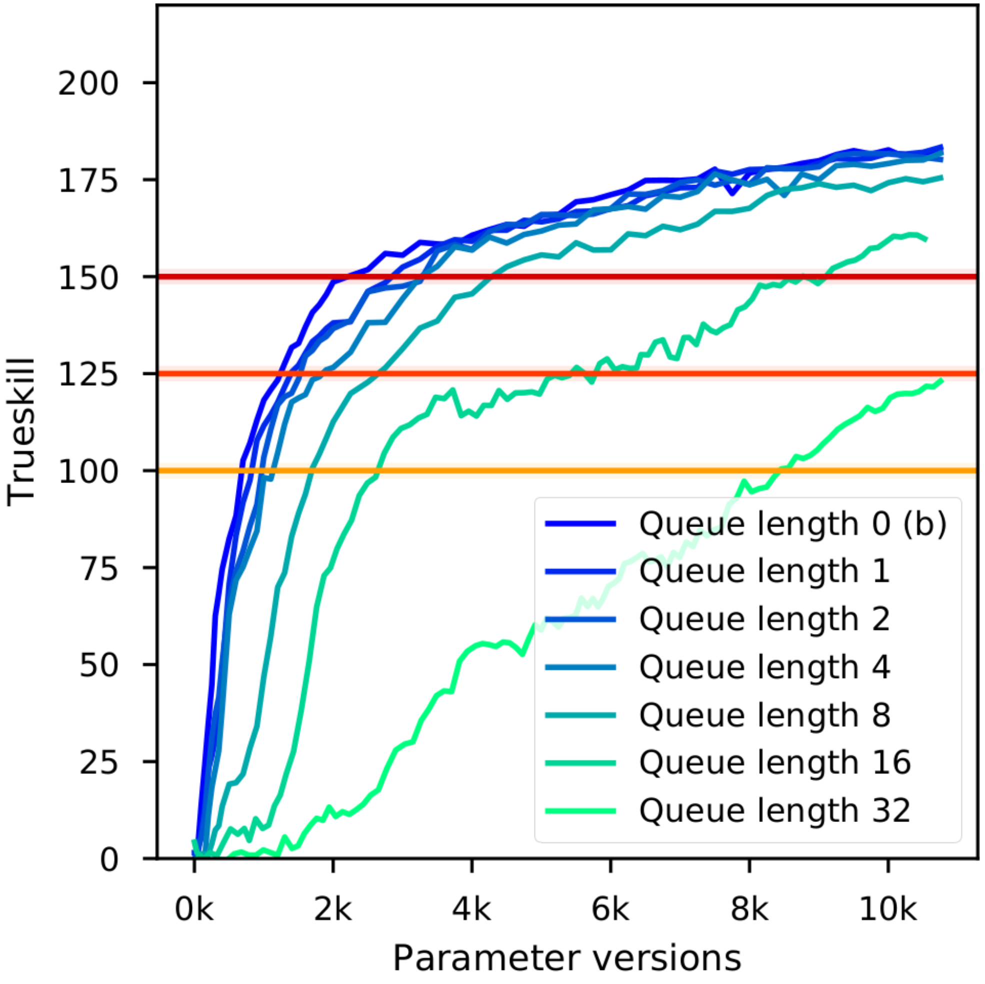
Next, we consider the operator $\mathcal{U}_2$ given by

$$\mathcal{U}_2 Q = Q + \alpha D_\rho \left(\mathcal{T}^* Q - Q\right), \tag{13}$$

where D_ρ is a diagonal matrix with entries $\rho(s, a)$, a probability mass function on state-action pairs.

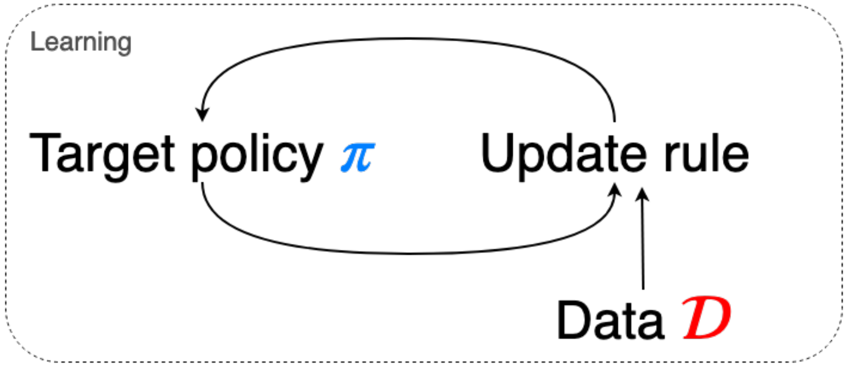
Lemma 2. *If $\rho(s, a) > 0$ for all s, a and $\alpha \in (0, 1/\rho_{max})$ where $\rho_{max} = \max_{s,a} \rho(s, a)$, then $\mathcal{U}_2$ given by Eq 13 is a contraction in the sup norm and its fixed-point is Q^*. If there are any s, a such that $\rho(s, a) = 0$ and $\alpha \in (0, 1/\rho_{max})$, however, it is a non-expansion in Q and not a contraction.*

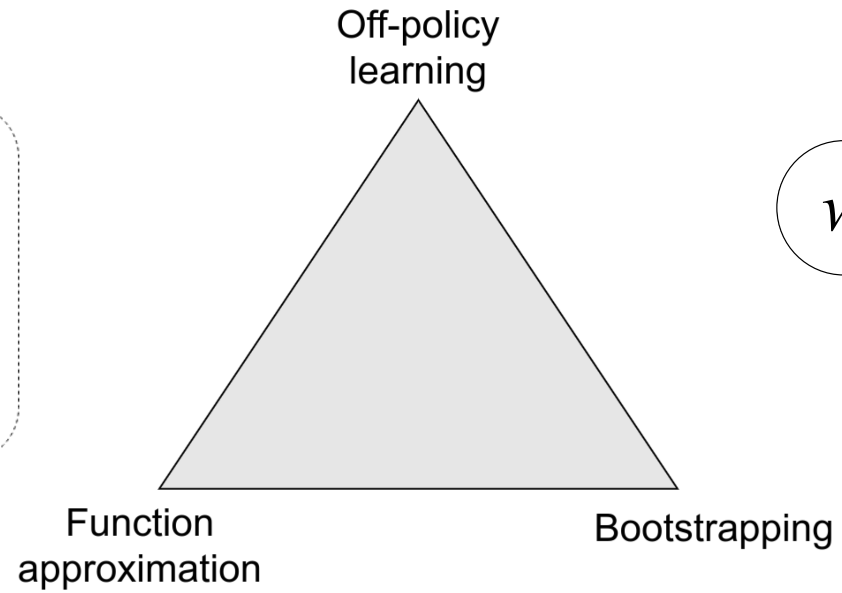
Increasing queue length of distributed PPO learning detrimental to performance in DOTA 2 (OpenAI, 2019. Dota 2 with Large Scale Deep Reinforcement Learning)

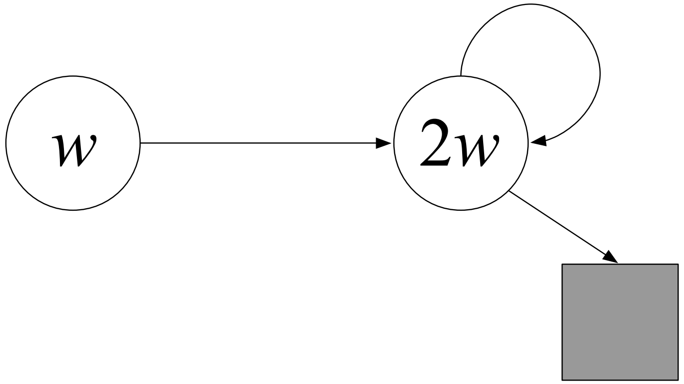


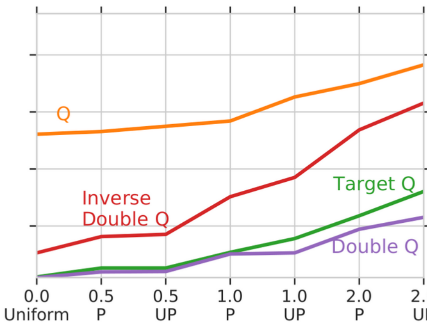
A3C with off-policy correction used for Starcraft II (DeepMind, 2019. Grandmaster level in Starcraft II using multi-agent reinforcement learning)

Summary

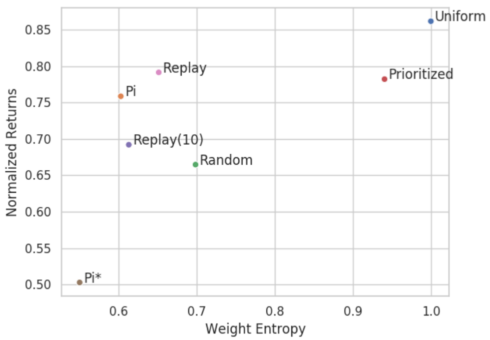








(c) Prioritization



Next, we consider the operator $\mathcal{U}_2$ given by

$$\mathcal{U}_2 Q = Q + \alpha D_\rho (\mathcal{T}^* Q - Q)\,, \tag{13}$$

where D_ρ is a diagonal matrix with entries $\rho(s,a)$, a probability mass function on state-action pairs.

Lemma 2. *If $\rho(s,a) > 0$ for all s, a and $\alpha \in (0, 1/\rho_{max})$ where $\rho_{max} = \max_{s,a} \rho(s,a)$, then $\mathcal{U}_2$ given by Eq 13 is a contraction in the sup norm and its fixed-point is Q^*. If there are any s, a such that $\rho(s,a) = 0$ and $\alpha \in (0, 1/\rho_{max})$, however, it is a non-expansion in Q and not a contraction.*