Reinforcement Learning

A3C / SIL

2021.03.31

최영제

Index

가치 함수와 벨만 방정식

- 상태 가치 함수(state value function)란 특정 상태에 도달 후 앞으로 얻을 수 있는 감가 누적 보상을 의미
- 행동 가치 함수(action value function, Q-function)란 특정 상태-행동을 취했을 때 앞으로 얻을 수 있는 감가 누적 보상을 의미

$$V_\pi(s) = E_\pi[G_t|S_t = s] \qquad Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$$

- 상태 가치 함수와 행동 가치 함수는 서로 변환이 가능하며 해당 성질을 활용하여 벨만 방정식을 도출할 수 있음

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \qquad Q_\pi(s,a) = E_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

- 벨만 방정식이란 현재 시점과 다음 시점의 가치 함수 간의 관계식을 말하며 가치 함수를 도출할 수 있는 식
- 벨만 최적 방정식은 최적 가치 함수를 도출할 수 있는 식이며 경험적인 방법을 통해서 풀어가야함(모델을 알 때 : DP / 모델을 모를 때 : MC, TD)
- 몬테 카를로 학습과 시간차 학습은 target을 무엇으로 두느냐에 따라서 차이가 존재함

MC와 TD

- 몬테 카를로 학습(MC-learning)은 감가 누적 보상(G_t)을 target으로 삼아 학습을 함

$$V(S_t) = V(S_t) + \alpha(G_t - V(S_t))$$

- 몬테 카를로 학습은 G_t를 사용하기 때문에 low bias / high variance 특징이 존재하며 episode가 끝나는 환경에서만 사용이 가능함(episodic)
- 에피소드가 끝날 때만 사용이 가능한 MC 대신에 시간차 학습(TD-learning)은 다음 시점의 가치 함수를 target으로 삼음

$$V(S_t) = V(S_t) + \alpha(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD-target}} - V(S_t))$$

TD-error

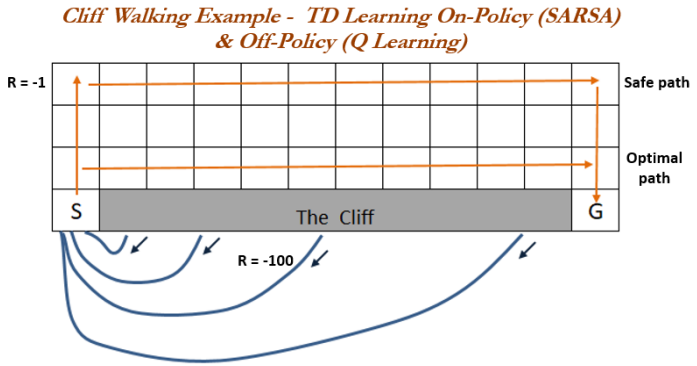
- 시간차 학습은 high bias / low variance 특징이 존재하며 episodic 환경이 아니더라도 활용이 가능함
- 이후 대부분의 알고리즘은 TD-learning을 기반으로 동작함

Q-learning

- Q-learning은 TD-learning 알고리즘의 하나로 행동 가치 함수를 학습하는 알고리즘, 갱신 공식은 다음과 같음

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{\acute{a}} Q(S_{t+1}, \acute{a}) - Q(S_t, A_t)]$$

- 앞의 TD-learning에서 다른 점은 행동 가치 함수를 학습한다는 것과 target으로 다음 시점의 maximum Q-value를 사용한다는 것
- Q-learning은 행동 정책(behavior policy)은 e-greedy를, 갱신 정책(target policy)는 greedy policy를 사용하는 off-policy 알고리즘임



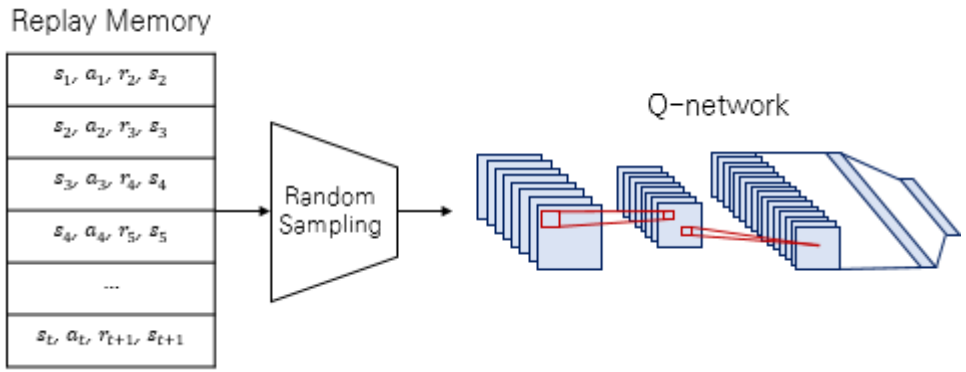
- 다만 Q-learning은 table 기반의 강화학습이기 때문에 state-action space가 크거나 무한한 공간에서는 적용할 수 없음
- 따라서 미분 가능한 함수(e.g., linear combination of features, neural net)를 이용하여 Q-value를 approximation하는 방법이 제안되었음

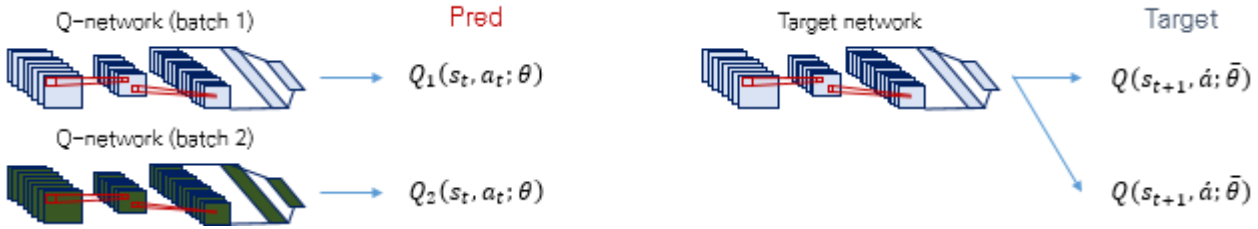
DQN

- DQN 이전 neural network를 강화학습에 적용한 approach는 다음과 같은 단점이 존재하였음

  1) Small volum of training data
  2) High correlation between samples
  3) Non-stationary target problem

- DQN은 experience replay와 fixed Q-target으로 위 문제를 해결하였음





PER

- Replay memory에는 좋은 transition과 그렇지 않은 transition이 혼재해 있음
- 따라서 n개의 transition을 random하게 sampling하는 DQN은 효율적이지 않음
- Prioritized experience replay는 이를 개선하여 좋은 transition에 높은 확률을 두어 sampling을 진행함
- 좋은 transition의 기준은 TD-error로 부터 도출되며 TD-error > priority > sampling probability의 과정을 거쳐 확률이 계산됨

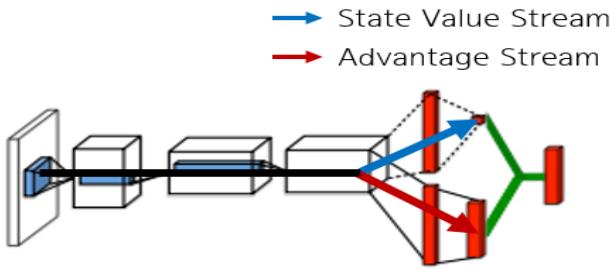
$$\delta_j = R_j + \gamma \underset{a}{max} Q(S_j, a) - Q(S_{j-1}, A_{j-1})$$

$$p_j \leftarrow |\delta_j| + \varepsilon$$

$$P(j) = \frac{p_j^\alpha}{\sum_k p_k^\alpha}$$

Dueling DQN

- DQN의 구조를 변경하여 성능 향상을 시도한 대표적인 예는 Dueling DQN이 존재함
- Dueling DQN은 state value와 advantage function을 추정 후 이를 합하여 Q-value를 근사함
- Advantage function은 Q-value와 state value 간의 차이로 정의되며 이를 통해 variance를 낮추는 효과가 있음



$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

- 다만 저자들은 두 가지 1) Q-value의 유도 경로를 알지 못하고 2) Q-value를 신뢰할 수 없다는 문제를 지적하여 다음과 같이 식을 변경함

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + [A(s,a;\theta,\alpha) - \max_{a\in\mathcal{A}} A(s,\acute{a};\theta,\alpha)]$$

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

- Dueling DQN의 실험 결과는 advantage function의 평균을 뺀 결과가 가장 좋았으며 이는 true advantage function의 기대값은 0이라는 성질을 활용

Policy gradient theorem

- Value based reinforcement learning은 위 알고리즘을 모두 포괄하며 가치 함수 갱신> 정책 갱신 > 가치 함수 갱신 > 정책 갱신 …의 과정을 거침
- Policy based reinforcement learning은 직접적으로 policy를 추정하고 학습하여 value function을 필요로 하지 않음
- 정책 신경망(policy network, π_θ)에 대해서 존재하는 모든 가중치 θ 중 가장 높은 보상을 주는 최적의 가중치 θ^*를 찾는 것이 목적임
- θ^*를 도출하기 위해서는 policy의 가치를 정의해야하며 다음과 같이 정의할 수 있음

  1) Episodic environment의 경우 초기 상태 가치 함수를 목적함수로 정의

  $$J(\theta) = V_{\pi_\theta}(s_1)$$

  2) Non-episodic environment의 경우 stationary distribution을 안다고 가정하면 모든 상태들의 평균 가치 함수로 정의

  $$J(\theta) = \sum_s d_{\pi_\theta}(s) V_{\pi_\theta}(s)$$

- 따라서 PG는 최적화 문제에 해당하며 목적 함수를 최대화 시키도록 gradient ascent를 사용함

$$\theta^* = \underset{\theta}{argmax} J(\theta)$$
$$\theta = \theta + \alpha \nabla_\theta J(\theta)$$

Policy gradient theorem

- 다만 위 식에서 한 가지 문제는 stationary distribution을 안다고 가정하였다는 것이며 이는 model based reinforcement learning에서만 활용 가능함
- Policy gradient theorem은 model free reinforcement learning 환경에서 사용하기 위해 위 식에서 $d_{\pi_\theta}(s)$를 정리하는 방법임
- 기댓값의 정의를 활용하여 다음과 같이 policy gradient를 변경하였음

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(A_t|S_t)Q_{\pi_\theta}(S_t, A_t)\right]$$

- 이를 통해 policy gradient에 stationary distribution에 대한 가정이 필요가 없으므로 model free reinforcement learning 환경에서 사용이 가능함
- Policy gradient에서 $\nabla_\theta log\pi_\theta(A_t|S_t)$ term은 score function이라 부르며 policy network의 output이 discrete한 환경이라면 cross entropy와 동일함
- Policy based reinforcement learning에서 policy network의 정답 vector는 agent가 행동했던 값을 1로 둔 one-hot vector임 (예 : [0,0,0,1])
- 해당 값만 활용을 하게 되면 agent는 기존의 행동만을 반복하게 됨 따라서 $Q_{\pi_\theta}(S_t, A_t)$ 값을 사용하여 update의 방향성을 제시함
- 만약 택한 행동이 좋았다면 기존의 one-hot vector와의 차이를 줄이도록 학습하고 좋지 않았다면 차이를 키우도록 학습을 진행함

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta log\pi_\theta(A_t|S_t)}_{\text{양}}\underbrace{Q_{\pi_\theta}(S_t, A_t)}_{\text{방향}}\right]$$

REINFORCE / Actor-Critic

- Policy based reinforcement learning algorithm들은 policy gradient에서 방향에 해당하는 Q-function을 어떤 것으로 두느냐의 차이임
- REINFORCE algorithm은 해당 Q-function을 단순하게 감가 누적 보상인 G_t로 대체한 것이며 policy gradient를 작성하면 다음과 같음

$$\nabla_\theta J(\theta) = \nabla_\theta log \pi_\theta(A_t|S_t) G_t$$

- REINFORCE는 G_t를 사용하기 때문에 monte carlo policy gradient라고도 불림
- 다만 G_t를 사용하기 때문에 episodic environment에서만 활용이 가능하며 variance가 크다는 단점이 존재함
- Actor-Critic algorithm은 원래의 policy gradient를 그대로 사용하고자 별도의 network를 두어 $Q_{\pi_\theta}(S_t, A_t)$를 추정하여 사용하고자 하였음
- 기존의 policy network는 actor network, Q-function을 추정하는 network는 critic network로 불러 actor-critic algorithm임
- Critic network의 학습 목표는 TD-error를 줄이는 것이며 DQN의 목적함수와 동일하게 TD-error의 제곱을 사용함

$$Loss_{critic} = [\mathrm{r} + \gamma Q_w(\dot{s}, \dot{a}) - Q_w(s, a)]^2$$

- TD-error를 사용하기에 TD 버전의 policy gradient 알고리즘이라고도 불리며 critic network를 먼저 학습하고 actor network를 나중에 학습함
- G_t를 사용하지 않기 때문에 non-episodic environment에서 활용이 가능하며 variance가 작은 장점이 있음

Baseline function

- Variance를 줄이는 또 다른 방법은 baseline function을 이용하는 것임 예를 들어 grid world의 Q-value가 [101, 99, 100, 104]라고 가정
- Q-value의 역할은 행동의 좋고 나쁨을 알려주는 것이기 때문에 Q-value의 평균인 101을 뺀 [0, -2, -1, 3]를 사용하여도 “Right” 행동이 가장 좋은 행동임을 알 수 있음 (variance를 줄이는 역할도 수행)
- 이 예에서 101과 같이 Q-value를 감해주는 값을 기준값 (baseline, $B(s)$)이라고 함
- 즉 policy gradient 식을 다음과 같이 사용하겠다는 것임

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s)\left(Q_{\pi_\theta}(s,a) - B(s)\right)\right]$$

- Baseline function은 action과 무관한 함수이면 어떤 함수를 사용해도 상관없음
- 다른 말로 action과 무관한 함수여야 아래 식이 성립하여 기댓값에 변화를 주지 않으면서 variance를 줄일 수 있음

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s)\left(Q_{\pi_\theta}(s,a) - B(s)\right)\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s)Q_{\pi_\theta}(s,a)\right]$$

Advantage Actor-Critic (A2C)

- A2C는 위 policy gradient에서 baseline function을 state value function으로 대체한 알고리즘을 말함

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s)\left(Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s)\right)\right]$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s) A_{\pi_\theta}(s,a)\right]$$

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

- 즉 value function을 추정하기 때문에 actor-critic이고, baseline을 뺀 식이 advantage function과 동일하여 위 식으로 가중치를 갱신하는 알고리즘을 advantage actor critic (A2C)라고 부르는 것
- 다만 위 식은 action value function과 state value function 두 가지를 추정해야 하기 때문에 다음과 같이 식을 변형

$$A_w(s,a) = r + \gamma V_w(\dot{s}) - V_w(s)$$

$$Q_{\pi_\theta}(s,a) = r + \gamma V_{\pi_\theta}(\dot{s})$$

- 또한 이 때의 advatnage의 형태는 state value에 대한 TD-error의 형태와 동일하여 최종적으로 gradient 공식은 다음과 같음

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(a|s)(r + \gamma V_w(\dot{s}) - V_w(s))]$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(a|s) A_w(s,a)]$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(a|s)\delta_w]$$

- Critic network의 목적 함수는 앞의 actor critic과 동일함

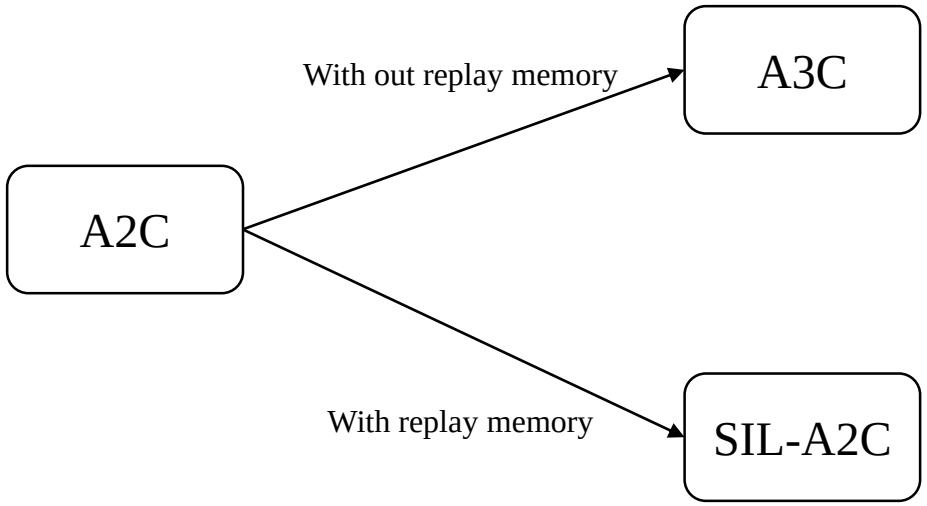
$$Loss_{critic} = [r + \gamma V_w(\dot{s}) - V_w(s)]^2$$

$$= [\delta_w]^2$$

Asynchronous Methods for Deep Reinforcement Learning (A3C)

A2C의 개선 알고리즘

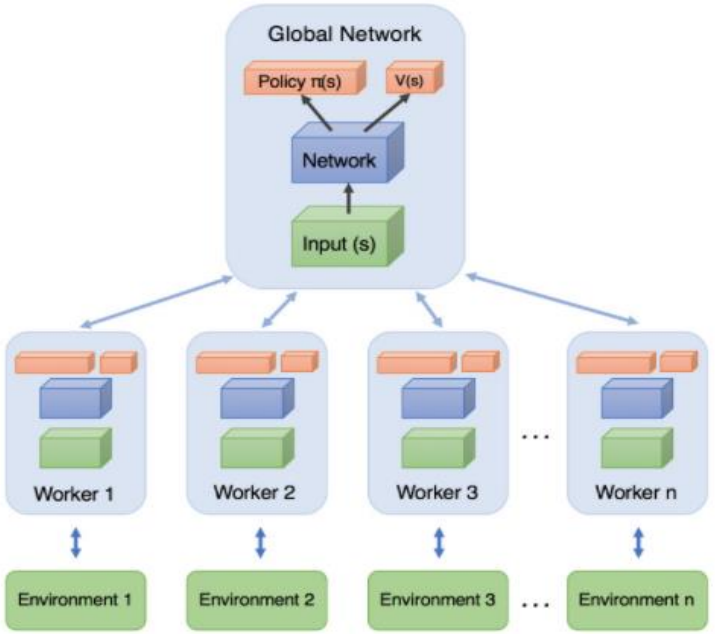
- Reinforcement learning의 training sample (transition)은 highly correlated sample에 해당하여 local minimum에 빠질 위험이 존재
- 이는 TD version의 policy based reinforcement learning인 A2C에도 동일하게 해당
- 따라서 이를 보완하기 위한 algorithm인 asynchronous advantage actor-critic (A3C)와 self imitation learning (SIL)에 대해서 알아볼 것



Asynchronous Methods for Deep Reinforcement Learning (A3C)

A3C

- Asynchronous advantage actor-critic algorithm은 replay memory 없이 다수의 agent를 구축하여 A2C의 단점을 개선하고자 하였음
- 다수의 agent들의 경험은 서로 연관되어 있지 않기 때문에 replay memory를 사용하지 않더라도 높은 상관성 문제를 해결할 수 있음
- 즉 A3C는 다수의 agent를 병렬적으로 구축하여 이들의 경험을 종합하여 높은 상관성 문제를 해결한 algorithm



Actor learner : A2C

Asynchronous Methods for Deep Reinforcement Learning (A3C)

Pseudo code of A3C

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Asynchronous Methods for Deep Reinforcement Learning (A3C)

Results of A3C

Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

Table 1. Mean and median human-normalized scores on 57 Atari games using the human starts evaluation metric. Supplementary Table SS3 shows the raw scores for all games.

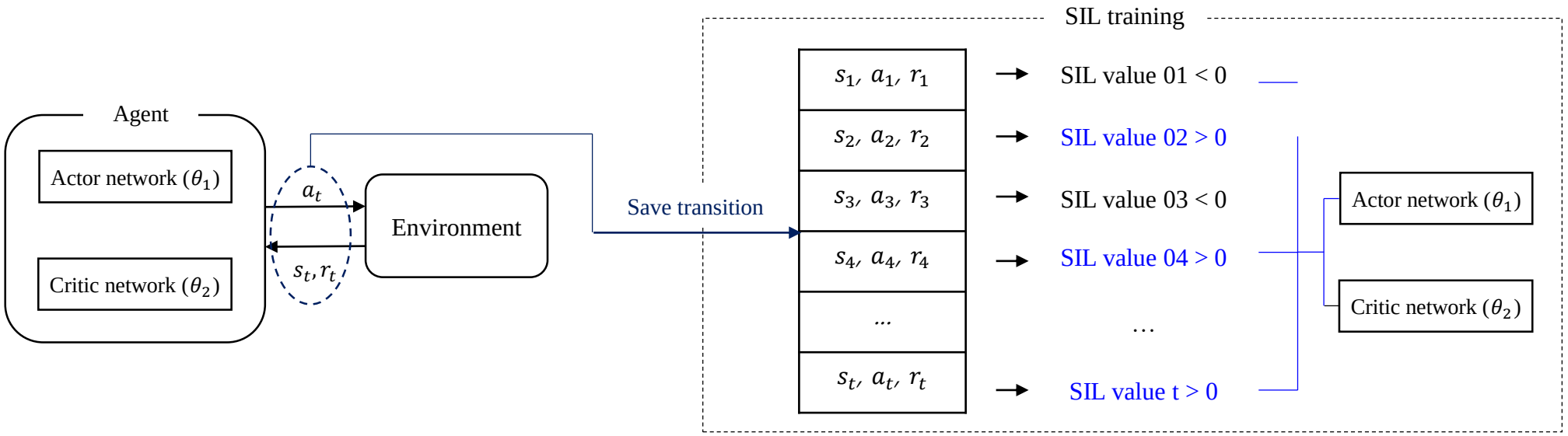
3 SIL Self imitation learning (SIL)

SIL

- A3C와 달리 replay memory를 사용하여 A2C를 개선한 algorithm을 self imitation learning (SIL)이라고 함
- SIL의 핵심 idea는 과거 agent의 경험한 transition을 memory에 저장 후 좋은 transition은 다시 사용하여 학습하는 것임(=모방)
- 좋은 transition의 기준은 episode 내 해당 time-step (t)의 discounted reward – state value function으로 정의 됨

$$Sil\ value = R_t - V_\theta(S_t) \qquad * R_t = \sum_k^\infty \gamma^{k-t} r_t \qquad \gamma \in [0,1]$$

- SIL framework를 사용하면 이 sil value값이 0보다 큰 transition은 다시 학습함



3 SIL Self imitation learning (SIL)

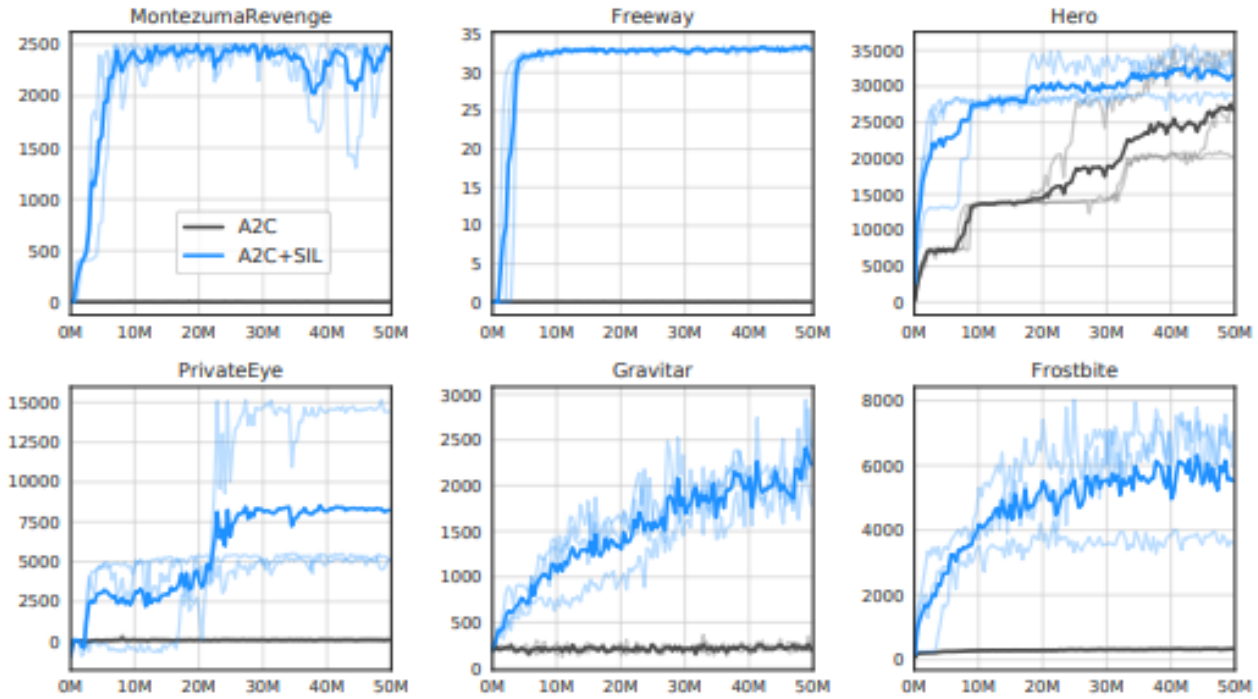
Pseudo code of SIL

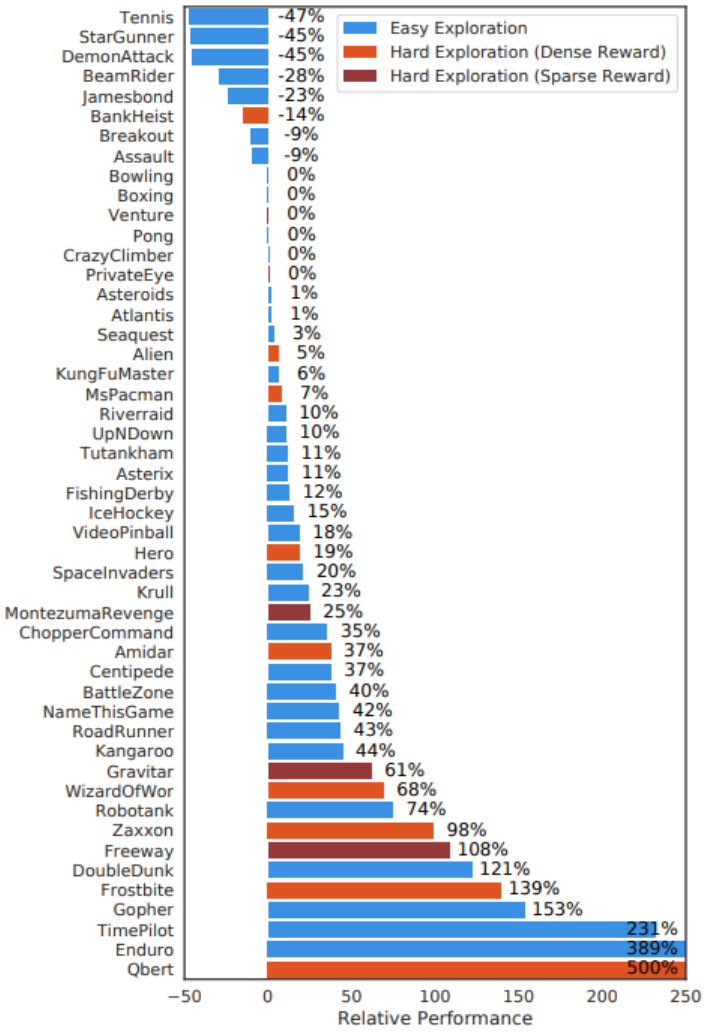
Algorithm 1 Actor-Critic with Self-Imitation Learning

Initialize parameter θ
Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
Initialize episode buffer $\mathcal{E} \leftarrow \emptyset$
for each iteration **do**
  # Collect on-policy samples
  for each step **do**
    Execute an action $s_t, a_t, r_t, s_{t+1} \sim \pi_\theta(a_t|s_t)$
    Store transition $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, r_t)\}$
  end for
  if s_{t+1} is terminal **then**
    # Update replay buffer
    Compute returns $R_t = \sum_k^\infty \gamma^{k-t} r_k$ for all t in $\mathcal{E}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R_t)\}$ for all t in $\mathcal{E}$
    Clear episode buffer $\mathcal{E} \leftarrow \emptyset$
  end if
  # Perform actor-critic using on-policy samples
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{a2c}$ (Eq. 4)
  # Perform self-imitation learning
  for $m = 1$ to M **do**
    Sample a mini-batch $\{(s, a, R)\}$ from $\mathcal{D}$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{sil}$ (Eq. 1)
  end for
end for

3 SIL Self imitation learning (SIL)

Results of SIL





Q&A