Reinforcement Learning

Policy Gradient / Actor-critic / A2C

2021.03.25

최영제

Index

1. Review

2. Policy gradient (REINFORCE / actor-critic)

3. Advatnage actor-critic (A2C)

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

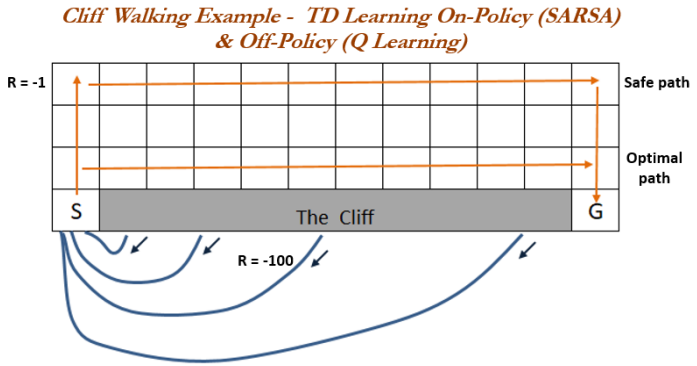
- 시간차 학습은 high bias / low variance 특징이 존재하며 episodic 환경이 아니더라도 활용이 가능함
- 이후 대부분의 알고리즘은 TD-learning을 기반으로 동작함

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



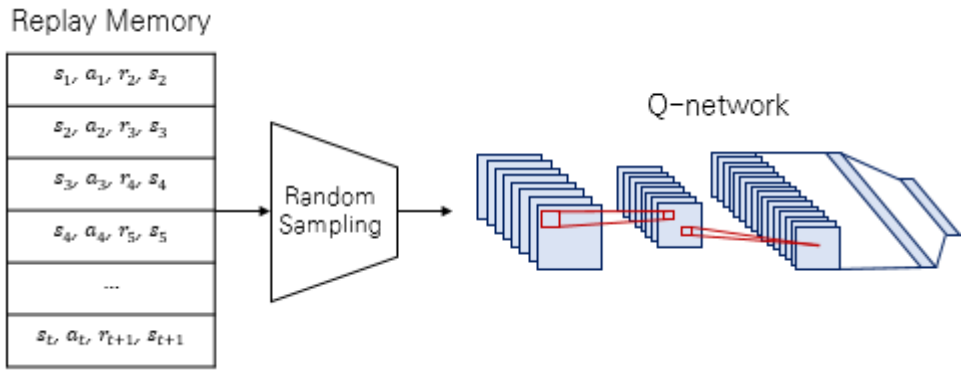
- 다만 Q-learning은 table 기반의 강화학습이기 때문에 state-action space가 크거나 무한한 공간에서는 적용할 수 없음
- 따라서 미분 가능한 함수(e.g., linear combination of features, neural net)를 이용하여 Q-value를 approximation하는 방법이 제안되었음

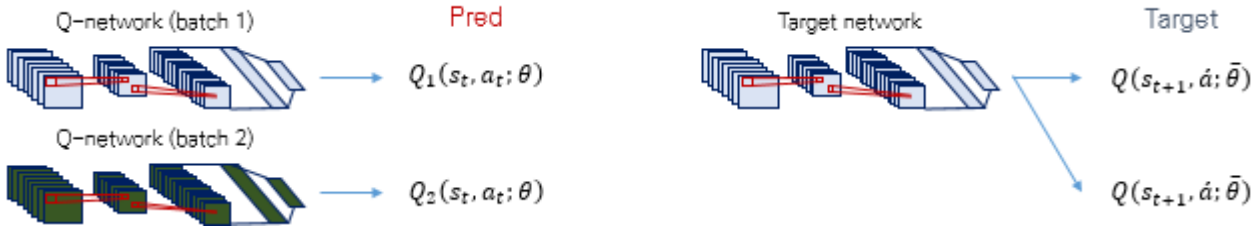
DQN

- 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

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

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

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

PER : simple example

- PerDQN을 위해서 각 transition 별 sampling probability를 구하면 다음과 같음(ε=0.01)

$$V(S) = (0,0,0,0,0)$$

$$\delta_7 = 1 + 0.9 \times 0 - 0$$
$$= 1$$

Replay Memory	TD-Error (δ)	Priority ($|\delta| + \varepsilon$)	Sampling Probability
T_1 — $< S_2, L, 0, S_1 >$	$\delta_1 = 0$	$p_1 = 0.01$	$P(1) = 0.0093$
T_2 — $< S_1, R, 0, S_2 >$	$\delta_2 = 0$	$p_2 = 0.01$	$P(2) = 0.0093$
T_3 — $< S_2, L, 0, S_1 >$	$\delta_3 = 0$	$p_3 = 0.01$	$P(3) = 0.0093$
T_4 — $< S_1, R, 0, S_2 >$	$\delta_4 = 0$	$p_4 = 0.01$	$P(4) = 0.0093$
T_5 — $< S_2, R, 0, S_3 >$	$\delta_5 = 0$	$p_5 = 0.01$	$P(5) = 0.0093$
T_6 — $< S_3, R, 0, S_4 >$	$\delta_6 = 0$	$p_6 = 0.01$	$P(6) = 0.0093$
T_7 — $< S_4, R, 1, S_5 >$	$\delta_7 = 1$	$p_7 = 1.01$	$P(7) = 0.9439$

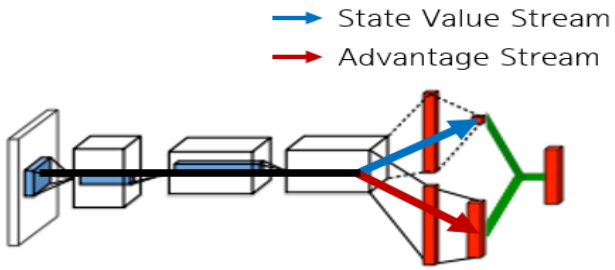
- Terminal state인 $V(S_5)$는 reward 값으로 갱신이 됨 $V(S_5) = 1$
- 위 확률을 따라 T_7이 뽑혔다고 가정 후 첫 번째 갱신을 진행하면 다음과 같음

$$V(S_4) = 0 + 1 \times (1 + 0.9 \times 0 - 0)$$
$$= 1$$

$$V(S) = (0,0,0,1,1)$$

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

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

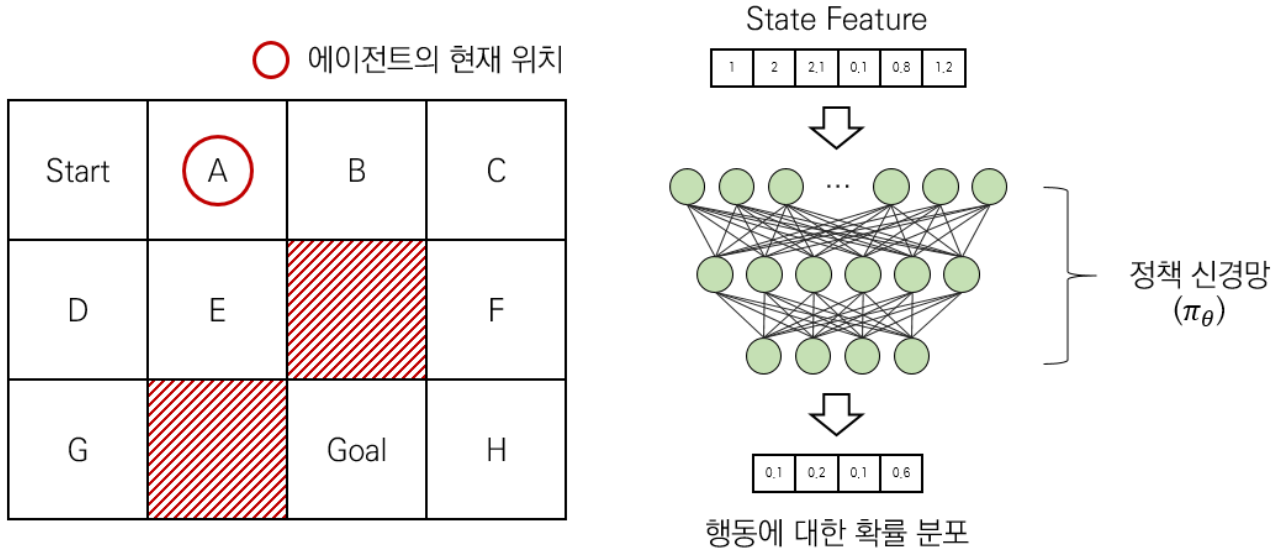
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + [A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{\acute{a}} A(s,\acute{a};\theta,\alpha)]$$

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

Policy Gradient

가치 기반 강화학습과 정책 기반 강화학습

- Q-learning, DQN 및 파생 DQN 알고리즘은 value-based reinforcement learning (가치 기반 강화학습)에 속함
- 가치 기반 강화학습은 가치 함수 추정(초기화) > 정책 갱신 > 가치 함수 추정 > 정책 갱신 … 의 과정을 거쳐 최적 가치 함수를 도출함
- Policy-based reinforcement learning (정책 기반 강화학습)은 이와는 달리 policy를 직접적으로 학습함
- Policy란 state를 입력으로 받아 action에 대한 distribution을 출력으로 삼는 함수로 정책 기반 강화학습에서는 가치 함수를 추정하지 않음
- Policy를 출력하는 network의 output인 action distribution에 따라 agent가 행동하고 이를 토대로 policy를 갱신하면 되기 때문



Policy Gradient

정책 기반 강화학습

- Policy-based RL의 목적은 정책 신경망(policy network, π_θ)에 대해서 존재하는 모든 가중치 θ 중 가장 높은 보상을 주는 최적의 가중치 θ^*를 찾는 것
- 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

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

- 다만 한 가지 문제가 존재하는데 목적함수를 정의함에 있어 stationary distribution을 안다고 가정하였다는 것
- 이는 환경에 대한 가정인 **모델을 알고 있을 때, 즉 model based reinforcement learning에서만 가능함**

Policy Gradient

Policy gradient theorem

- Model free reinforcement learning을 위해서는 policy gradient $\nabla_\theta J(\theta)$를 구할 때 $d_{\pi_\theta}(s)$에 의존적이지 않도록 변형해야 함
- 목적함수를 먼저 변형하면 다음과 같음

$$V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$$

$$J(\theta) = \sum_s d_{\pi_\theta}(s) V_{\pi_\theta}(s) \longrightarrow J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) Q_{\pi_\theta}(s,a)$$

- 변형된 목적함수의 policy gradient $\nabla_\theta J(\theta)$를 구하면 다음과 같음

$$\nabla_\theta J(\theta) \propto \nabla_\theta \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) Q_{\pi_\theta}(s,a)$$

$$= \sum_s d_{\pi_\theta}(s) \sum_a Q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s)$$

확률 / 값

- 변형된 policy gradient에서 $d_{\pi_\theta}(s)$를 제거하기 위해 식을 기대값으로 변형함 / 기대값 = ∑ (확률 * 값)

$$\mathbb{E}[X] = \sum_i p_i x_i$$

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

- 식이 기대값의 형태로 바뀌면서 기존 s를 확률 변수 S_t로 표기함

2 SIL Policy Gradient

Policy gradient theorem

- Policy gradient $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_a Q_{\pi_\theta}(S_t, a)\,\nabla_\theta \pi_\theta(a|S_t)\right]$ 중 $\nabla_\theta \pi_\theta(a|S_t)$ term을 다음과 같이 수정

$$\nabla_\theta \pi_\theta(a|S_t) = \pi_\theta(a|S_t)\frac{\nabla_\theta \pi_\theta(a|S_t)}{\pi_\theta(a|S_t)}$$

$$= \pi_\theta(a|S_t)\nabla_\theta log\pi_\theta(a|S_t)$$

$\frac{\nabla_\theta \pi_\theta(a|S_t)}{\pi_\theta(a|S_t)}$의 형태는 log함수를 미분한 수식과 동일

- 변형된 term을 토대로 다시 한번 policy gradient를 정리하면 다음과 같음

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_a \underbrace{\pi_\theta(a|S_t)}_{\text{확률}}\underbrace{\nabla_\theta log\pi_\theta(a|S_t)Q_{\pi_\theta}(S_t, a)}_{\text{값}}\right]$$

- 위 식은 다시 한번 기대값의 형태로 변환이 가능하며 기대값 안의 기대값은 기대값과 동일한 성질을 이용하여 다음과 같이 변환 가능함

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

- 이와 같은 과정을 통해서 우리는 model에 대한 가정 $d_{\pi_\theta}(s)$ 없이 policy gradient를 구할 수 있으며 이를 policy gradient theorem이라고 함
- 즉 policy gradient theorem을 통해서 우리는 model-free policy based reinforcement learning이 가능해짐

2 SIL Policy Gradient

Policy gradient theorem

- Policy gradient (PG)는 기존 model-base RL에서 벗어나 model-free RL에서도 policy based RL을 진행할 수 있다는 것에 큰 의의가 있음
- 중요한 것은 도출된 policy gradient가 무엇을 의미하는지, 결론적으로 policy gradient 알고리즘의 target vector는 무엇인지를 아는 것
- 먼저 policy gradient의 $\nabla_\theta log\pi_\theta(A_t|S_t)$ term은 score function이라 부르며 neural network에 적용될 gradient 값임
- 만약 policy network의 output이 discrete한 경우라면 softmax의 미분값이 이에 해당하며 손실함수로 cross entropy가 사용됨
- Cross entropy는 정답 vector와 예측 vector를 필요로 하며 예측 vector는 policy network의 output에 해당함
- 정답 vector는 agent가 실제로 택한 행동이 1의 값을 갖는 one hot vector임
- 가령 grid world에서 agent가 오른쪽으로 가는 행동을 할 경우 [0, 0, 0, 1]의 값이 정답 vector가 됨
- 위 방식으로 학습할 경우 과거의 같은 행동만을 반복하기 때문에 Q-value를 곱한 값을 gradient로 사용하는 것임
- 즉 좋지 않은 행동의 경우 음수의 Q-value 값을 통해 했던 행동이 좋지 않음을 알려주는 역할을 함

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

양 방향

Policy Gradient

REINFORCE

- 그런데 policy gradient 식을 살펴보면 Q-value값이 사용됨 $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(A_t|S_t)Q_{\pi_\theta}(S_t, A_t)]$
- 이는 초반에 가치 함수는 추정하지 않아도 된다는 policy based reinforcement learning의 내용과 맞지 않음
- 따라서 Q-value를 사용하지 않고 감가 누적 보상 (G_t)으로 이를 대체한 알고리즘이 REINFORCE 알고리즘임

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

- 여기서 기대값은 표본 추출을 통해 계산할 수 있는 값이므로 이를 제거하고 REINFORCE의 gradient를 구하면 다음과 같음

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

- 즉 REINFORCE의 가중치 갱신 공식은 다음과 같음

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

- REINFORCE는 G_t를 사용하기 때문에 monte carlo policy gradient라고도 불림
- 다만 G_t를 사용하기 때문에 episodic environment에서만 활용이 가능하며 variance가 크다는 단점이 존재함

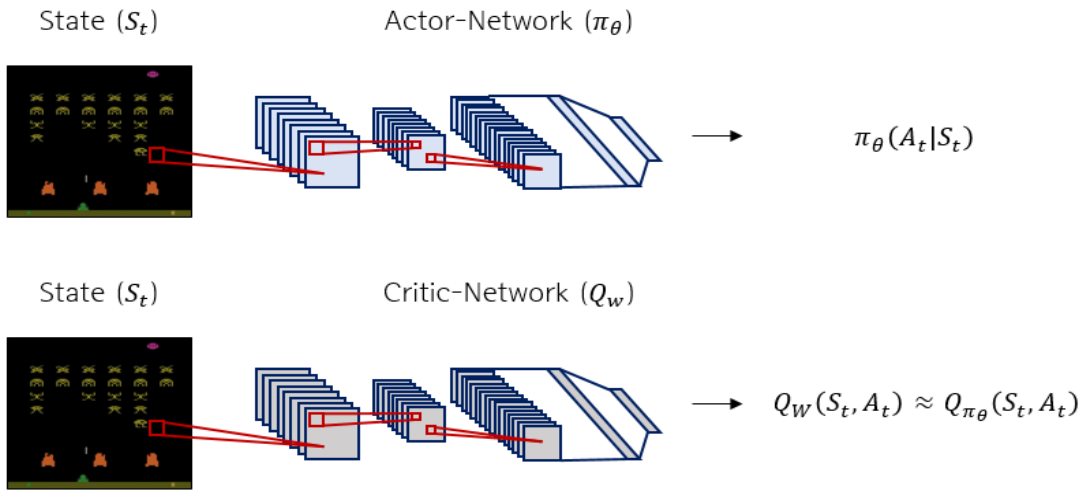
Policy Gradient

Actor-critic

- Non-episodic environment에서 활용이 가능하고 variance를 줄이기 위해서는 G_t를 사용하지 않으면 됨
- 따라서 $Q_{\pi_\theta}(S_t, A_t)$를 추정하고 원래의 policy gradient 식을 사용하는 방법이 등장함

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

- 즉 policy를 근사하는 network와 함께 value function을 근사하는 network를 도입한 알고리즘이 actor-critic임
- Actor는 policy를 critic은 value function을 추정하는 역할을 함



Policy Gradient

Actor-critic

- Critic network는 action value function을 추정할 수도 있고 state value function을 추정할 수 있음
- Critic network의 학습 목표는 TD-error를 줄이는 것이며 DQN의 목적함수와 동일하게 TD-error의 제곱을 사용함

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

- Actor critic 알고리즘의 동작 순서는 다음과 같으며 TD-error를 사용하기에 TD 버전의 policy gradient 알고리즘이라고도 불림

1) 상태 s, Actor-Network 의 가중치 θ, Critic-Network 의 가중치 w를 초기화

2) 상태 s와 Actor-Network π_θ를 사용하여 a를 샘플링$(a \sim \pi_\theta(a|s))$

3) 매 Time-Step 에 대해서

   A. 다음 상태 $\acute{s}$, 보상 r, 그리고 다음 시점의 행동 $\dot{a}$을 샘플링

   B. Critic-Network 의 업데이트를 위한 TD-Error(δ)를 계산

$$\delta = \mathrm{r} + \gamma Q_w(\dot{s}, \dot{a}) - Q_w(s, a)$$

   C. Critic-Network 업데이트

$$\mathrm{w} = \mathrm{w} + \beta\delta\nabla_w Q_w(s, a)$$

   D. 추정된 $Q_w(s, a)$를 이용하여 Actor-Network 업데이트

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

- G_t를 사용하지 않기 때문에 non-episodic environment에서 활용이 가능하며 variance가 작은 장점이 있음

Advantage actor-critic (A2C)

Baseline

- 다만 일반적으로 사람들이 말하는 actor critic은 advantage actor critic을 말함
- A2C는 기존의 actor-critic 알고리즘에서 variance를 더 줄이기 위하여 식을 수정한 알고리즘임
- 예를 들어 grid world의 Q-value가 [101, 99, 100, 104]라고 가정할 때, 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

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

- 이제 이렇게 변형을 해도 문제가 없는지 증명하겠으며 우선 다음과 같이 두 가지 항으로 분리할 수 있음

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

- 우리는 앞에서의 변환 과정($\sum_s d_{\pi_\theta}(s)\sum_a Q_{\pi_\theta}(s,a)\nabla_\theta\pi_\theta(a|s) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\pi_\theta(a|s)Q_{\pi_\theta}(s,a)\right]$)을 통해 뒤의 항은 다음과 같음을 알 수 있음

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

Advantage actor-critic (A2C)

Baseline

- Baseline $B(s)$는 **action에 대한 함수가 아니니 밖으로 뺌 (baseline으로 사용하려면 action에 무관한 함수를 사용해야함)**

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(a|s)B(s)] = \sum_s d_{\pi_\theta}(s)B(s)\nabla_\theta \sum_a \pi_\theta(a|s)$$

행동에 대한 확률분포의 합 = 1

- 즉 상수에 대한 미분값이므로 0의 값을 가짐

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(a|s)B(s)] = \sum_s d_{\pi_\theta}(s)B(s)\nabla_\theta 1$$

$$= 0$$

- 따라서 다음과 같은 식이 성립함

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

- 위 식의 의미는 action에 무관한 함수를 Q-value에서 감하더라도 기대값은 동일하다는 것 (**기대값의 변화는 없으면서 variance는 줄일 수 있음**)

Advantage actor-critic (A2C)

Baseline

- 그런데 행동 가치 함수에서 baseline을 뺀 식의 형태는 앞에서 Dueling DQN에서 확인한 바 있음

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

- State value function은 action과 무관한 함수이기 때문에 baseline으로 적합함
- 따라서 baseline function으로 state value function을 사용하면 policy gradient식은 다음과 같음

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

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

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

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

3 SIL Advantage actor-critic (A2C)

Actor network의 갱신 공식

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

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

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

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

- A2C의 actor network의 가중치 갱신 공식은 다음과 같음

$$\theta = \theta + \alpha\nabla_\theta log\pi_\theta(a|s)(r + \gamma V_w(\dot{s}) - V_w(s))$$

$$= \theta + \alpha\nabla_\theta log\pi_\theta(a|s)\delta_w$$

- Critic network의 목적 함수는 앞의 actor critic과 동일함

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

$$= [\delta_w]^2$$

Q&A