SAC는 이전 강화학습 모델들인 policy gradient와 DDPG, PPO의 문제점을 해결한 모델.

Off policy 이면서 동시에 Continuous Action space에서도 동작하는 알고리즘

Soft Actor-Critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor

강화학습의 목표는 어떤 환경이 있다면, 환경과 상호작용을 하는 에이전트가 존재. 에이전트는 환경으로부터 어떤 State를 받고 action을 환경에게 취함. 이에 환경은 에이전트에게 reward를 주고 새로운 state를 준다. 이후 리워드를 계속 증가시키는 것이 목적. 이 리워드의 합들을 return이라고 하는데, 이 return을 maximize하는 최선의 policy, action들을 뽑아낼 수 있게 하는 것이 강화학습의 목적.

Value-Based: return의 기댓값이 output으로 반환이 되고, 리턴의 기댓값이 큰 쪽으로 액션을 선택

Policy-Based:: return의 클 확률이 마지막 신경망의 아웃풋으로 나타나고 이 확률을 다시 샘플링해 어떤 액션을 취할지 샘플링한다.

SAC에서는 어떻게 더 나은 모델을 만들 수 있었을까?

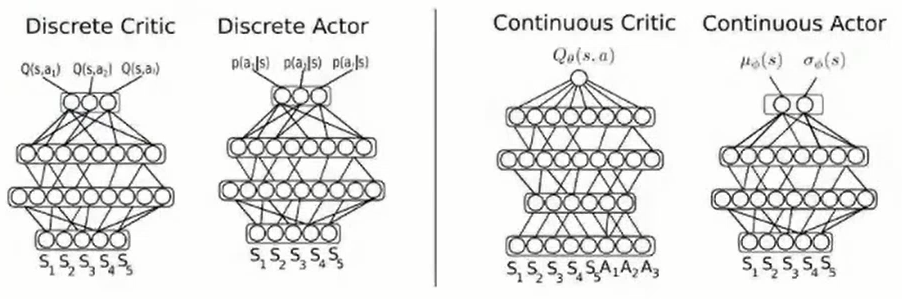
1. Stochastic Actor

텍스트이(가) 표시된 사진

자동 생성된 설명

세타는 가상의 축과 막대의 각도. 따라서 여기서 reward는 세타의 크기와 막대의 각속도의 합을 마이너스로 취하는 것이므로, 세타가 최대한 작고, 좌우로 움직이지 않고 가만히 있을 때 가장 높은 리워드를 취한다는 것. 여기서 문제는 왼쪽이나 오른쪽의 이산적인 문제가 아니라, 얼마나 각으로 힘을 줄 것인지 -2~2까지의 값 중 하나를 stochastic output으로 설정하는 것.

Cartpole은 classification 문제였으면, 이는 regression 문제.



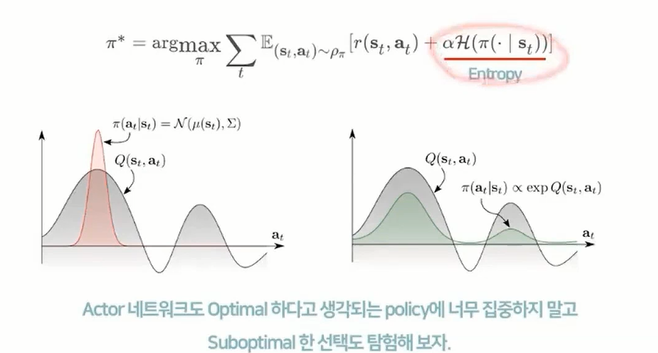
왼쪽은 cartpole, 오른쪽은 pendulum. 각각의 액션에 대해서 아웃풋이 나올 때, Continuous 에서는 state와 action에 대해 평가하는 값이 하나 나오고, u와 sigma를 리모델링 함으로써 매번 새로 값이 샘플링 하는 것. 이를 stochastic actor라고 한다. Actor는 실제로 어떤 에이전트를 어떻게 움직이게 할지 라는 policy라고 말할 수 있다.

1. Critic

하나의 에피소드가 끝날 때까지 state와 action, 그에 따른 리워드를 저장 해둔 후, 에피소드가 끝나면 리워드가 컸던 액션의 확률을 좀 더 크게 만들고, 그 액션을 한 번 더 수행했는데 다음번에 리워드가 조금 작게 들어온다면 그러면 그 전에 내가 한 액션을 조금 작게 만들도록 액션을 훈련시키는 자는 아이디어가 강화학습을 위한 policy gradient 방법론에 제일 초기의 아이디어. 이를 reinforce라고 한다. 이 reinforce라는 알고리즘은 매번 하나의 에피소드에 1 번의 gradient step만을 할 수 있고, 따라서 어떤 정책이 에피소드가 끝날 때까지는 무조건 그 policy를 따라 움직여야 하기 때문에 variance가 크다. 따라서 각각 액션이 좋은지 나쁜지 바로 바로 확인할 수 있는 다른 네트워크를 훈련 시키자라는 아이디어가 있었는데, 이 다른 네트워크를 critic이라고 한다. Actor network와 critic network 존재. GAN 생각 이 두가지 네트워크를 따로따로 훈련시켜야 강화학습 알고리즘이 동작할 수 있다.

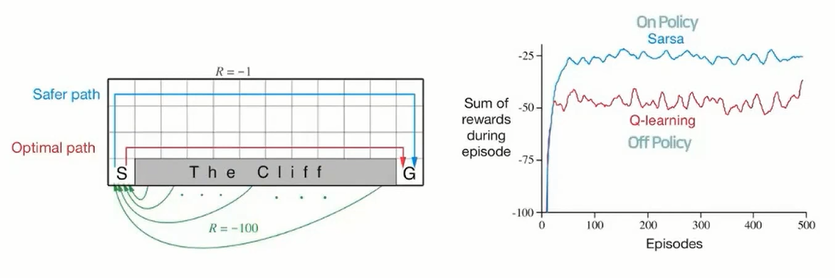
1. Maximum Entropy

엔트로피를 maximize한다는 것. 엔트로피는 확률변수가 가지고 있는 정보량. 따라서 actor의 엔트로피도 최대한 증가시키고, 크리틱의 엔트로피도 증가시키는 것. 이전에는 크리틱의 네트워크는 리턴의 기대값을 평가할 때 조금 낙관적으로, 후하게 평가를 하는 경향이 있었으니까 엔트로피항을 추가하여 하나로 고집을 하지 말고 그걸 맞춰보자는 것.



이와 같이 actor에서도 엔트로피항을 추가. 엔트로피가 없으면 일반적인 Actor critic 알고리즘이 되는 것. 예를 들면 정책(파이)을 normal distribution으로 모델링을 한다면 어떤 크리틱이 액션에 따라 예상치를 왼쪽처럼 한다고 했을 때, 최대한 리워드가 큰 쪽에 대해서 액션을 취하려고 한다. 그러나 엔트로피항을 추가하여 정책을 훈련시킨다면 크리틱이 평가하는 거에 비례를 해서 어떤 액션을 고를지로 바뀜. 이때도 물론 최대한 큰 Q에 대하여 최대한 방향을 따라가는 것은 맞지만, Sub optimal하다고 생각되는 액션도 Stochastic actor로 인해 적당한 확률로 뽑히므로 이 쪽도 고려를 하게 된다. 내가 단순히 현재 알고 있는 지식 중에서 가장 좋다고 생각되는 방향 뿐 아니라 다른 방향도 탐색하여 조금 더 나은 방법이 있는지 탐색하게 된다.

1. Off-policy



Cliff 문제. 왼쪽은 S에서 시작하여 G로 가는 것을 목표로 하는 알고리즘. 상하좌우로 움직일 때마다 -1의 reward를 받는다. 이때, Cliff로 가게 된다면 -100점을 얻는다.

On-policy 알고리즘은 에이전트가 행하는 것이 최적의 policy이기 때문에 최대한 안전하게 가려고 한다. 반면 off-policy는 behavior policy가 따로 있기 때문에 target policy는 영향을 당장에는 조금 덜 받게 된다. 결국 exploration을 진행하며 optimal policy 찾도록 조금 더 노력하는 것. 이때 그래프를 보면 on policy인 sarsa가 더 좋다고 볼 수 있지만, off-policy는 위험에도 불구하고 위험 주변에서 계속 exploration을 진행한다는 것이므로 조금 더 지나면 off-policy 알고리즘은 Q-learning이 훨씬 높은 reward를 받게 될 것. 물론 on-policy 알고리즘인 sarsa 역시 높은 reward를 받는 방향(최적의 해)을 찾게 될 것이지만, Q-learning보다 훨씬 많은 episode 진행해야 할 것.

결론적으로 on-policy는 작은 negative한 reward에 민감하게 반응하여 학습을 진행하여 큰 목적(reward)을 달성하기 까지 오랜 시간이 걸릴 수 있다.

반면 off-policy는 조금 더 위험한 선택을 할 수 있다. 작은 Negative한 reward를 반복해서 수행하고, 이 명령을 데이터화 시켜 계속 학습시킬 수 있다. 이렇게 모인 데이터들을 통하여 이 데이터들을 계속 reuse 함으로써 target policy라고 하는 최적의 policy를 학습시킬 수 있으므로, 조금 더 데이터 샘플링에 efficient하다라고 생각할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 전체 for 문들은 episode에 대해서 계속 얼만큼의 업데이트를 진행 할 것인지 말하는 것 텍스트이(가) 표시된 사진

자동 생성된 설명

이 부분은 어떤 behavior policy를 취해서 계속해서 샘플링을 하여 데이터 안에 데이터를 모으는 단계라는 것. 이때 D는 Store the transition in the replay pool.

텍스트이(가) 표시된 사진

자동 생성된 설명

Store된 D를 통해서 다양한 데이터와 저장된 D를 통하여 Policy들이 폴리엑터와 크리틱들이 업데이트 되는 것을 확인 할 수 있다. 결국 데이터를 모으는 단계와 데이 터로 신경망 엑터와 크리틱을 업데이트 하는 두 가지 단계로 나누어져 있다.