Overcoming Exploration in Reinforcement Learning with Demonstrations

Ashvin Nair12, Bob McGrew1, Marcin Andrychowicz1, Wojciech Zaremba1, Pieter Abbeel12
International Conference on Robotics and Automation(2018)

IIE8557-01 동적계획법과 강화학습

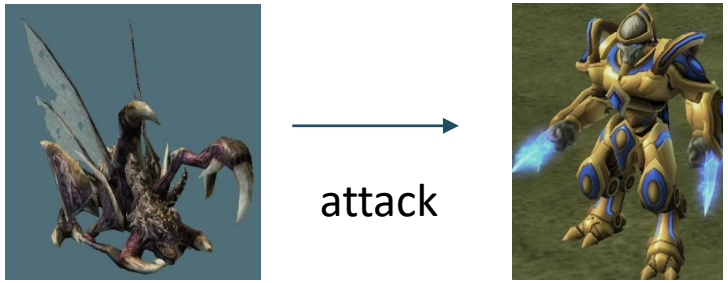
경영과학연구실 김변민

Sparse Reward Environment

- Agent 가 보상을 얻는 상황이 희박한 환경
- 특정 도메인에서 에이전트가 목표에 도달하는데 필요한 과정이 복잡해지면 Reward 가 발생하지 않는 (s,a) 가 많아 질 수 있음
- Sparse Reward Environment 에서는 Exploration 을 통해서는 보상을 받기 힘듬

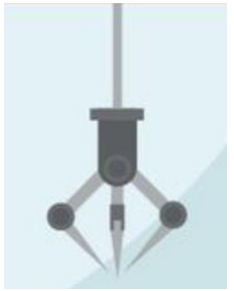
EX) Game domain vs Robot domain

Game domain



공격하는 순간 Reward 발생

Robot domain



물건을 A -> C 로 옮기면 reward 발생하고 그 과정에서는 reward 가 없음

Spare reward 로 인해 발생하는 문제를 해결하는 기법을 전문가 시연을 이용하여 개발하고자 함

Key Idea

- 시연을 강화학습에 효과적으로 접목 시킴으로써 무작위 exploration 단계를 단축 시킴

  - 시연에서 도착한 state를 추출하여 이를 agent 의 초기 state로 선정

  - Reward 가 발생하지 않는 rollout 에도 Reward 를 발생시키도록 조정

- Q-filter 를 이용하여 학습된 정책이 시연보다 더 나을 때 시연의 영향을 줄임

Behavior Cloning Loss

- Behavior Cloning Loss란 전문가 정책과 agent 정책의 차이를 손실함수로 표현한 것
- 정책을 업데이트하는 과정에서 Behavior Cloning Loss를 보조 손실로 사용함

정책 파라미터 업데이트 과정 (정책 기반 강화학습)

$$\theta_\pi = \theta_\pi + \alpha(\lambda_1 \nabla_{\theta_\pi} J - \lambda_2 \nabla_{\theta_\pi} L_{BC})$$

→ 평가함수 J 를 최대화함과 동시에 손실을 최소화(전문과 액션과의 차이)하는 방향으로 정책 파라미터가 업데이트 됨

$$\lambda_1 \nabla_{\theta_\pi} J - \lambda_2 \nabla_{\theta_\pi} L_{BC}$$

보조손실

Gradient ascent

Gradient descent

$$L_{BC} = \sum_{i=1}^{N_D} \|\pi(s_i|\theta_\pi) - a_i\|^2$$

전문가의 action과 agent 의 action의 차이를 이용한 Loss (일반적으로 많이 쓰임)

Qfilter

- 전문가 정책이 항상 옳지는 않을 수 도 있음
- 크리틱 Q(s,a) 를 이용해 Demonstration action이 agent action 보다 좋은 지 판단함

$$L_{BC} = \sum_{i=1}^{N_D} \|\pi(s_i|\theta_\pi) - a_i\|^2 \quad \Rightarrow \quad L_{BC} = \sum_{i=1}^{N_D} \|\pi(s_i|\theta_\pi) - a_i\|^2 \, \mathbb{1}_{Q(s_i,a_i)>Q(s_i,\pi(s_i))}$$

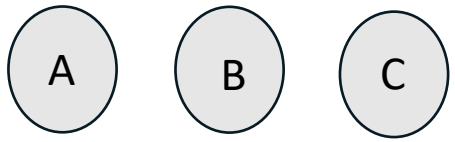
전문가 액션의 Q값이 agent action의 Q 값보다 클 때만 손실함수 적용

HER

- HER(Hindsight Experience Replay) 이란 ?

HER 의 아이디어 : 보상이 발생하지 않은 실패한 rollout에서도 보상을 받을 수 있도록 변환시킴

EX) 로봇 팔



Port

- 기존 목표는 A -> B 로 옮기는 것이었지만 결과는 A->C 가 되어버림
- 한번의 rollout을 아래와 같이 버퍼에 두 번 넣음

  - 실패한대로 (목표 : A->B 실제 결과 : A->C)

  - 원래 목표가 A- > C 로 옮기는 것이었다고 가정함 (목표 : A->C 실제 결과 : A->C)

- 실패한 상황에서도 Reward 를 받을 수 있게 함으로서 Sparse reward 상황에서의 학습을 보강함

Reset to demonstration states

- Reset to demonstration states

아이디어 : HER 의 아이디어를 transfer learning 도메인에 접목시킴

EX) 로봇 팔

A B C

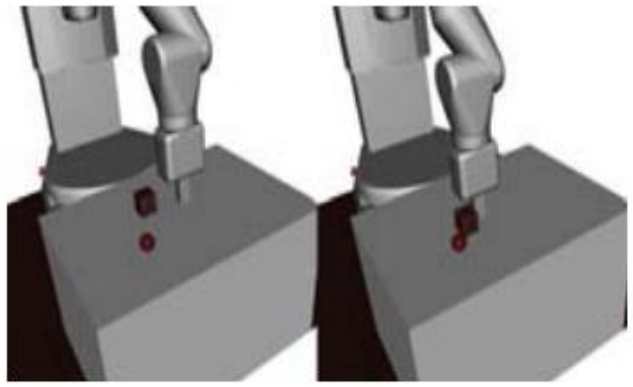
Port

- 에이전트는 전문가가 기존에 방문하였던 state 를 초기 상태로 설정하고 해당 demonstration에서 실제로 도달한 최종 상태를 목표로 함
- 샘플링한 전문가의 rollout : A -> C
- 에이전트 시작 (state = A, 목표 = C)
- 실제로 에이전트가 A -> B 로 간다면 목표를 재 수정하여 buffer 에 넣음 (HER 적용)

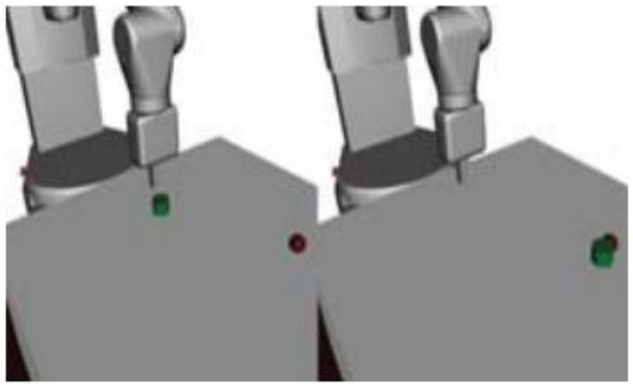
  - 실패한대로 (목표 : A->C 실제 결과 : A->B)
  - 원래 목표가 A- > B 로 옮기는 것이었다고 가정함

로봇으로 블록 옮기기

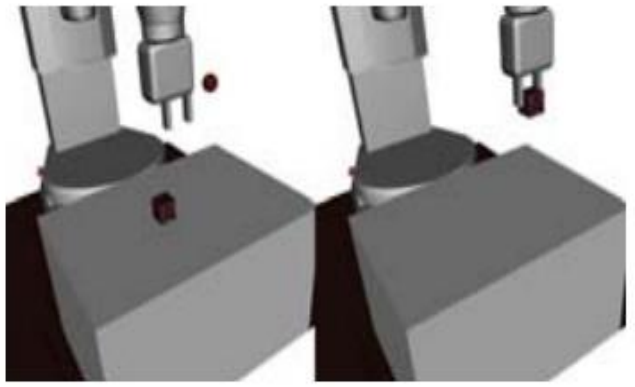
- Pushing : 로봇이 블록을 밀면서 목표지점으로 이동시킴
- Sliding : 로봇이 블록을 한번에 튕겨서 목표지점으로 이동시킴
- Pick and place : 로봇이 블록을 집어서 목표지점으로 이동시킴



Pushing



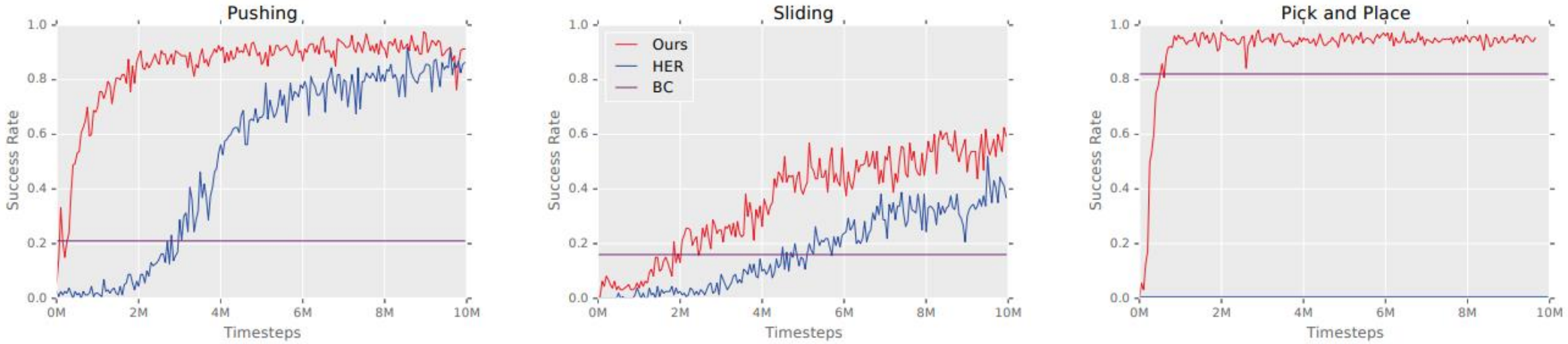
Sliding



Pick and Place

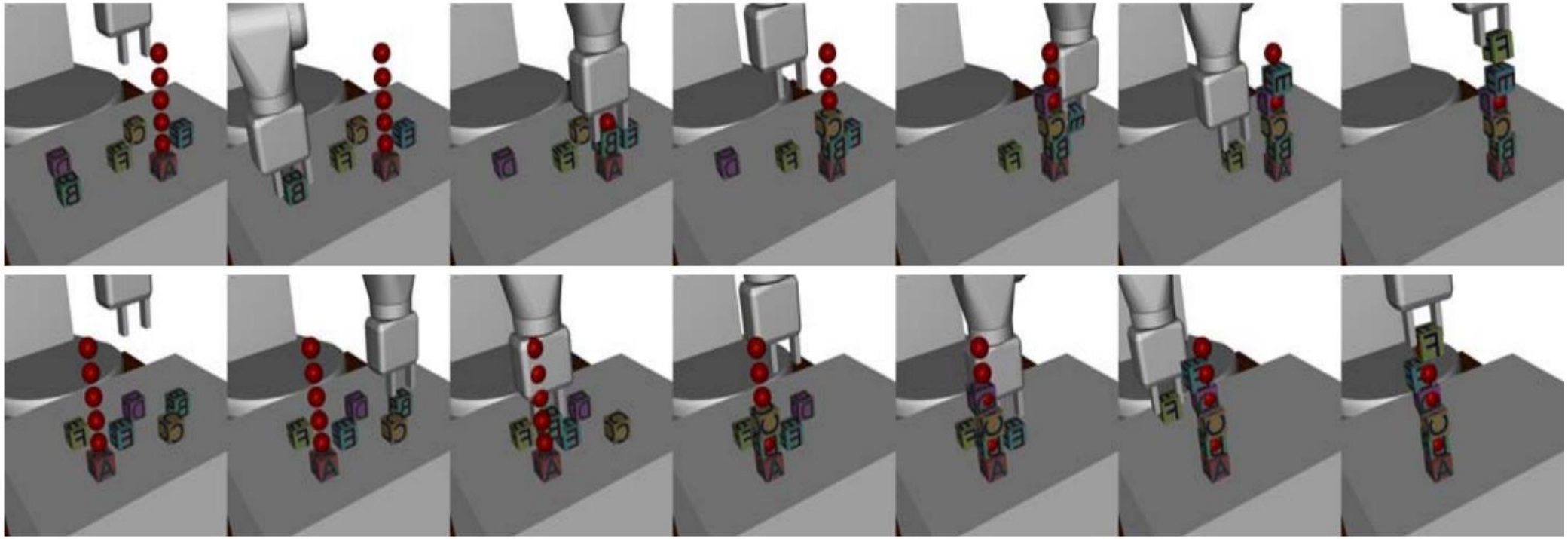
로봇으로 블록 옮기기 결과

- Ours : 제안한 방법에서 Reset to demonstration states를 사용하지 않은 것
- Pick and place 에서는 우연히라도 로봇이 블록을 잡을 수 없기에 HER 방법이 한번도 성공하지 못함



로봇으로 블록 쌓기

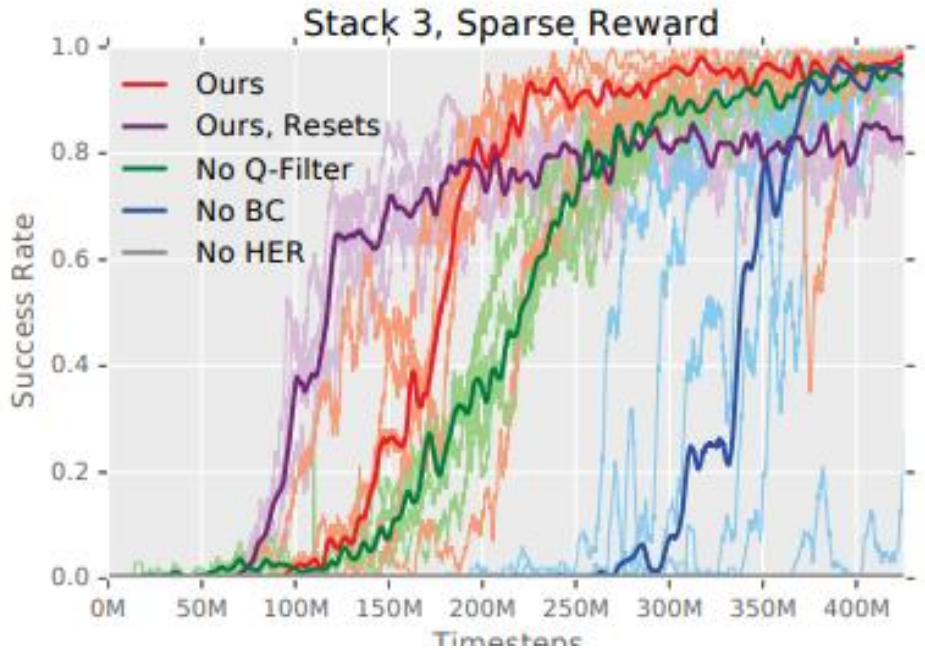
- Pick and place 를 이용하여 블록 탑이 무너지지 않게 쌓아야 함
- Sparse reward 가 심한 상황과 아닌 상황으로 나누어서 실험을 진행



로봇으로 블록 쌓기 결과

- 제안한 방법이 전체적으로 기존 방법보다 좋은 성능을 보임
- 작업량이 늘고 보상이 희소해질 때 Resets 를 포함하면 성능이 대폭 향상됨
- 쌓아야 할 블록수가 많아질수록 Reset 사용이 효과적임

Task	Ours	Ours, Resets	BC	HER	BC+ HER
Stack 2, Sparse	99%	97%	65%	0%	65%
Stack 3, Sparse	99%	89%	1%	0%	1%
Stack 4, Sparse	1%	54%	-	-	-
Stack 4, Step	91%	73%	0%	0%	0%
Stack 5, Step	49%	50%	-	-	-
Stack 6, Step	4%	32%	-	-	-



Conclusions

- 제안한 방법은 성공기준을 명확히 정할 수 있는 상황에서 보편적으로 적용 가능함
- Demonstration 을 이용했을 때와 아닐때 명확한 성능차이를 보임
- 이를 실제 환경에 적용하기 위해서는 많은 양의 샘플이 필요하지만 이를 구하기 어렵다는 한계점이 있음

Q & A