DQN 구조

> Author : nemo
> 2020-08-31

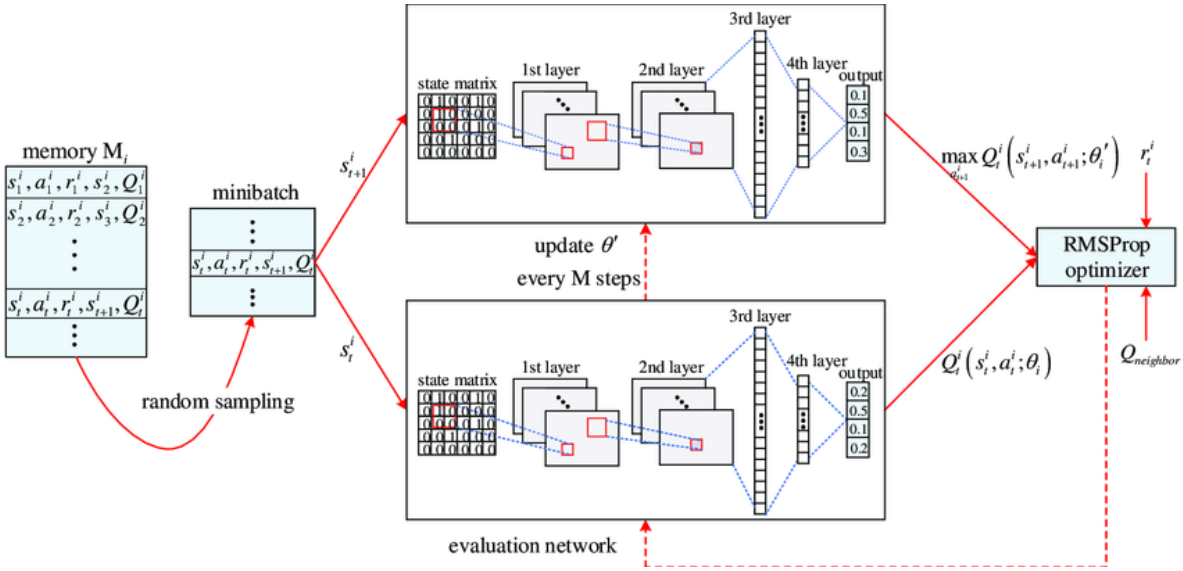
Contents

지난 글에서 Q-Learning의 수렴성에 대한 내용을 다루었으며, Bellman Optimality Equation에 대한 이해도를 높였다.

이번 글에서는 DQN에 관한 이론과 구조에 대해 알아보자.

DQN을 짜기 위해 Q-Learning의 Q를 신경망으로 바꾸면 ... 어림도 없지 !! 학습이 잘 되지 않는다. 신경망 근사로 인해 수렴성이 깨지기 때문이다. 학습이 잘 되게 하려면 크게 두 가지(Experience Memory, Target Network)의 추가적인 구현이 필요하다.

각각을 왜 구현해야하는지, 어떻게 구현해야 하는지 알아보자.



1. Replay Memory

> Replay Memory는 직전의 경험이 현재의 Q에 주는 영향을 줄이기 위해 사용한다.

유사한 상태에 대한 행동 가치는 유사할 것이다. Q를 신경망에 근사하면 이런 부분을 반영하기 때문에 메모리 측면에서 효과적인 것이다.

하지만 신경망을 근사하는 과정에서 이는 방해된다. 현재 상태는 이전 상태와 유사한데, 이전 상태를 즉시 업데이트 하면, 현재 상태로 업데이트할 때 영향을 받는다.

에피소드를 진행하는 과정에서 얻은 새로운 경험을 State, Action, State_Next, Reward으로 이루어진 데이터로 만들어, Replay Memory에 쌓아두었다가 이후 랜덤으로 뽑아 신경망을 훈련시킨다.

보통 Replay Memory의 크기는 1e4 이상의 값을 사용하며, 랜덤으로 뽑는 Batch의 크기는 32~128 정도로 한다.

이를 통해 두 가지 이익을 얻을 수 있다

- 직전의 학습이 현재의 환경에 끼치는 영향이 감소한다.
- 순서를 랜덤하게 하여 데이터 간 시간적 연관성을 줄였다.

2. Network Speration

2.1. Loss Function (Bellman Error)

함수 근사를 하려면 손실함수가 필요하다. 이전 글에서 설명한 Bellman Error를 손실함수로 사용한다.

> $r(s,a) + \gamma max_a Q(s',a) - Q(s,a)$

이를 밸만 에러(Bellman error)라 하는데, 오차니까 최소화하는 방향이 좋으며 손실함수로 사용할 수 있다.

제곱해서 MSELoss로 써도 되고 절댓값 함수만 씌워도 돈다.

2.2. Network Seperation

> Network Seperation는 직전의 Network Update가 현재의 Q에 주는 영향을 줄이기 위해 사용한다.

2013 논문과 2015논문의 차이점이다. 없어도 학습은 된다.

하나의 네트워크만 이용해서 업데이트 하는 경우, 업데이트 직후 Loss Function에 주는 영향이 커져서 학습이 불안정해지기 때문에, 타겟 값을 계산할 때 가중치를 고정시킨 Target Network를 이용하며, 학습은 Main Network만 시키고, 만 스텝 등 충분히 긴 시간 간격을 두고 Target Network를 Main Network의 가중치로 업데이트 한다.

3. 기타

3.1. History

Atari-BreakOut 환경은 한 프레임에 대한 State를 반환하는데, 한 프레임으로는 공의 속도나 방향에 대한 정보를 얻을 수 없다.

따라서 4개의 프레임을 하나의 State로 사용하자.

다음 글에서 Atari-BreakOut에 대한 실습을 통해 위의 동작을 자세히 이해해보자.

Reference

DQN - 블로그