Prioritized Experience Replay

AI 키워드	
날짜	@2024년 11월 11일
콘텐츠	논문
태그	유런

💡
- 2015
- Google DeppMind
- DQN의 uniformly sampled experience replay를 중요도에 따라 prioritized experience replay로 바꾸어서 성능 향상

Experience Replay의 도입

Q-learning

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

π: 정책(policy), Q(s,a)를 최대로 하는 action을 찾는 식. (π 옆에 붙은 *는 최적의 값임을 말하는 데 Q를 최대로 하는 action을 취하는 정책이 최적임을 의미)

딥러닝과 강화학습을 결합하고자 하는 시도

기존 Q learning은 Q(s, a)를 테이블 형식으로 저장하여 학습 → 이를 비선형 함수로 근사

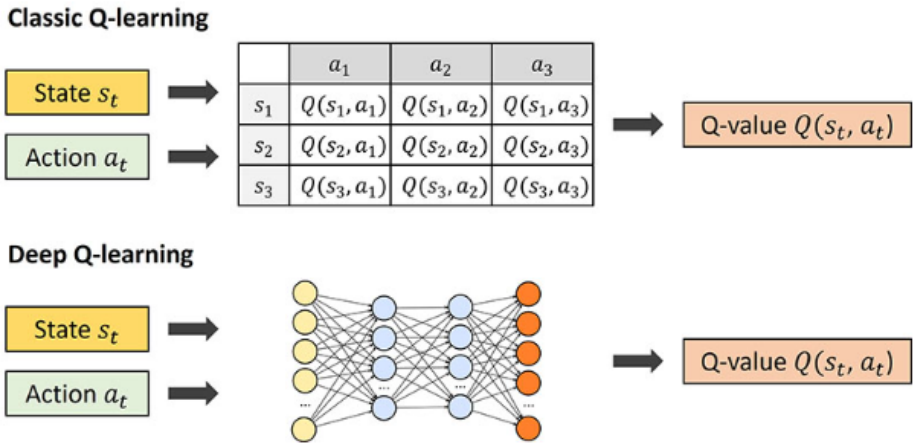


Figure 1. Q-learning과 deep Q-learning 비교

문제점

sample correlation

data distribution 변화

움직이는 target value

DQN

experience replay와 target network를 도입하여 문제점 해결

[기존의 Deep Q-learning algorithm]

1) 파라미터를 초기화하고, 매 스텝마다 2~5를 반복한다.

2) Action a_t 를 ϵ-greedy 방식에 따라 선택한다.

3) Action a_t 를 수행하여 transition $e_t = (s_t, a_t, r_t, s_{t+1})$ 를 얻는다.

4) Target value $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$ 를 계산한다.

5) Loss function $(y_t - Q(s_t, a_t; \theta))^2$ 를 최소화하는 방향으로 θ 를 업데이트한다.

DQN에서는 3번 과정에서 experience replay가, 4~5번 과정에서 target network가 적용됨

experience replay

[Experience Replay]
1) 매 스텝마다 추출된 샘플 $e_t = (s_t, a_t, r_t, s_{t+1})$ 을 replay memory D 에 저장한다.
2) Replay memory D 에 저장된 샘플들을 uniform하게 랜덤 추출하여 Q-update 학습에 이용한다.

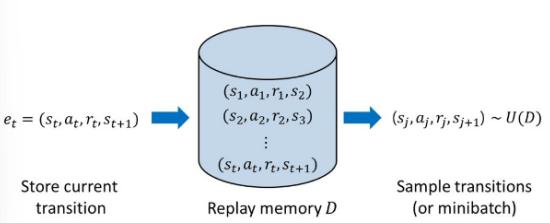


Figure 2. Experience replay

샘플 e_t를 바로 평가에 이용하지 않고 의도적으로 지연시

target network

[Target network]
1) Target network θ^- 를 이용하여 target value $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-)$ 를 계산한다.
2) Main Q-network θ 를 이용하여 action-value $Q(s_j, a_j; \theta)$ 를 계산한다.
3) Loss function $(y_j - Q(s_j, a_j; \theta))^2$ 이 최소화되도록 main Q-network θ 를 업데이트한다.
4) 매 C 스텝마다 target network θ^- 를 main Q-network θ 로 업데이트한다.

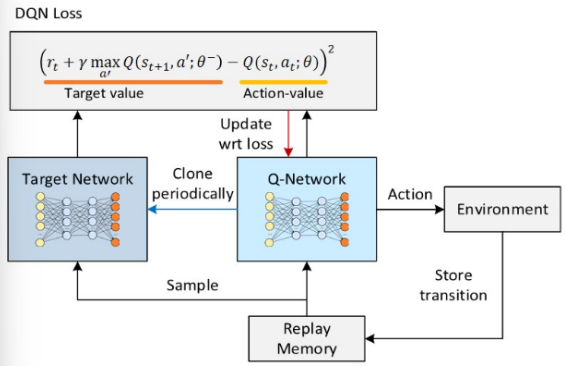


Figure 4. DQN의 target network 구조

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

Figure 5. DQN 알고리즘 (2015 Nature)

INTRODUCTION

DQN은 uniform 하게 샘플을 랜덤추출 하는데 이는 더 좋은 데이터와 덜 좋은 데이터의 차이를 구분하지 못하는 단점이 있기 때문에 prioritized experience replay를 만듦

이 논문에서는 temporal-difference(TD) error의 크기를 기준으로 priority를 정함

TD error: value function의 예측값과 실제값의 차이

BACKGROUND

뇌과학 실험에서 쥐의 뇌가 experience replay를 사용하고 있음이 밝혀짐

  보상과 관련된 sequence 더 자주 replay

  TD error가 높은 sequence 더 자주 replay

이를 강화학습 알고리즘에 적용한 prioritized sweeping이라는 기법과 TD error를 이 priority의 기준으로 적용한 연구 존재

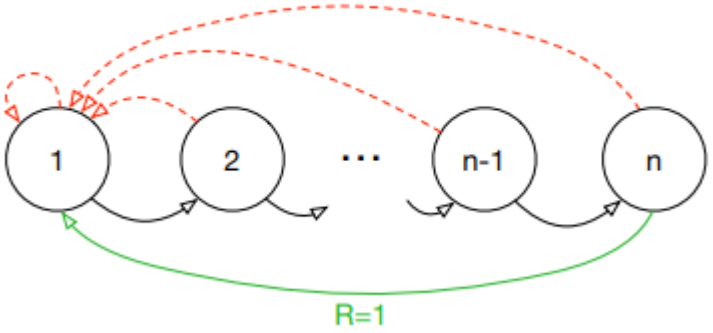
이 논문은 이 아이디어를 기반으로 model-free RL에 맞추고, stochastic sampling 사용

PRIORITIZED REPLAY

replay memory를 사용하기 위해 고려해야할 두 가지: 어떤 experience를 저장할까, 어떤 experience를 학습할까 → 이 논문에서는 후자만 고려

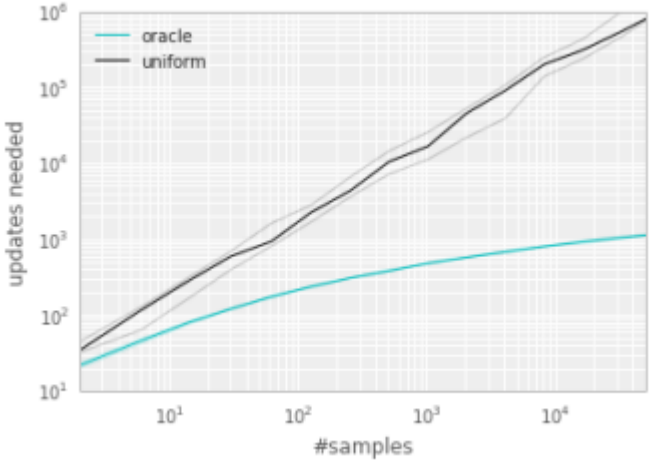
A MOTIVATING EXAMPLE

Blind Cliffwalk environment: 보상이 rare할 때는 학습이 어려움을 보여줌. uniform하게 random으로 뽑히는 experience를 사용한다면 대량의 실패에 성공한 일부 experience가 묻힐 것(=선택되지 않을 것)



Blind Cliffwalk

실제로 어떠한 전략을 가지고 experience를 선택하는 것이 더 빠른 학습 속도를 보여줌



PRIORITIZING WITH TD-ERROR

TD error를 기준으로 우선순위를 정하면 현재 얼마나 더 학습해야하는 지 알 수 있음

online RL에는 적합하지만(SARSA, Q learning) reward가 noisy한 경우에는 잘 작동하지 않을 수 있음

Blind Cliffwalk 환경에서 greedy TD error prioritization로 학습 시켰을 때 효과가 나타남

STOCHASTIC PRIORITIZATION

greedy TD error prioritization의 문제점

- TD error 순서대로 replay가 되므로 TD-error이 낮은 sequence는 한 번도 replay가 안될 수 있음
- stochastic reward 같은 noise에 취약
- 너무 한정된 experience만 반복해서 학습

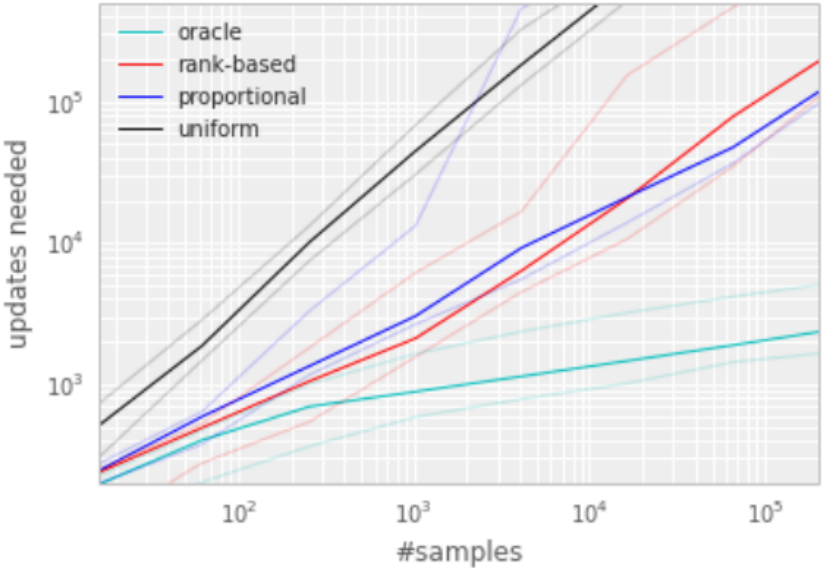
→ **stochastic sampling** : priority는 유지되지만 모든 experience에 대해 0이 아닌 샘플링 확률이 보장됨(=한 번도 안 뽑힐 일은 없음)

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

p: priority, α: 얼마만큼의 prioritization을 사용하는지에 대한 hyperparameter (α=0일 때 uniform)

이 논문에서는 p_i에 대해 두 가지 방식 사용

1. Proportional prioritization: $p_i = |\delta_i| + \epsilon$ (ϵ은 0이 되는 것 피하기 위함)
2. Rank-based prioritization: $p_i = \frac{1}{\text{rank}(i)}$



둘 다 monotonic(비례)이지만, rank-based가 이상치에 덜 민감하기 때문에 더 robust할 것

Algorithm 1 Double DQN with proportional prioritization

Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

ANNEALING THE BIAS

prioritization을 사용하면 expected distribution에서 멀어지기 때문에 bias가 생김 → 수렴해야 하는 정답에서 멀어짐 → **importance sampling** 가중치 도입

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$$

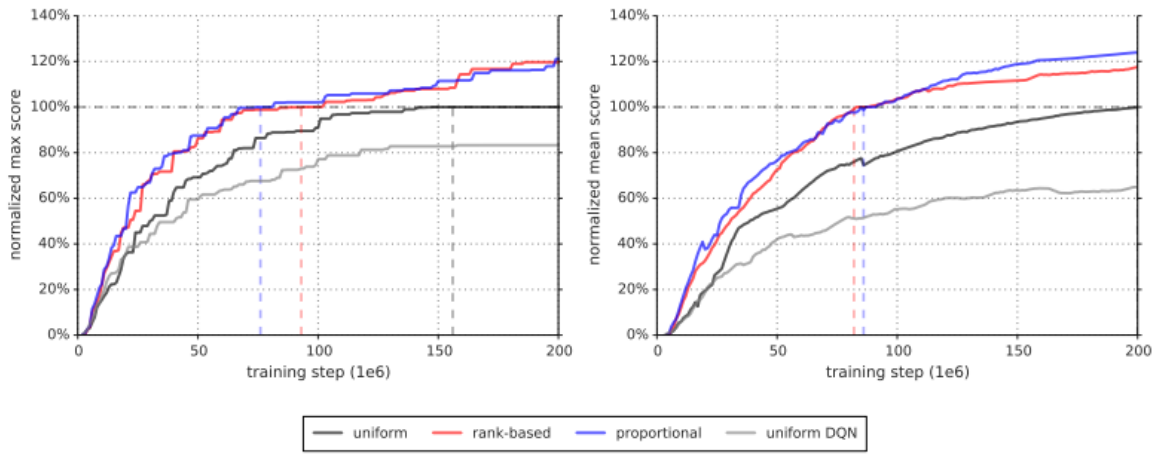
Q-learning update의 δ_i 대신 $w_i\delta_i$로 사용($\frac{1}{\max_i w_i}$로 normalize)

β를 어떤 값에서 시작하여 1로 만들면(=annealing) 처음에는 크게 update되다가 점점 안정적으로 변함 → error가 높은 experience를 자주 보게 만들면서 그에 따른 gradient magnitude를 줄일 수 있음

ATARI EXPERIMENTS

Atari benchmark에서 DQN과 DDQN에 PER을 결합하여 SOTA 달성

hyperparameter인 α와 β_0은 coarse grid search를 사용하여 적절한 값 탐색



	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
Median	48%	106%	111%	113%	128%
Mean	122%	355%	418%	454%	551%
> baseline	–	41	–	38	42
> human	15	25	30	33	33
# games	49	49	57	57	57

Table 1: Summary of normalized scores. See Table 6 in the appendix for full results.

DISCUSSION

rank-based가 더 robust할 것이라고 예상했지만 실제로는 별 차이 없었음. DQN 알고리즘에서 보상과 TD error의 클리핑이 아웃라이어를 제거하는 효과를 주기 때문일 것.

TD 오류의 분포가 학습이 진행됨에 따라 점차 중간값이 커지는 경향이 있음

일부 experience는 메모리에서 나오기 전까지는 replay되지 는 현상이 발견됨(다른 experience는 긴 지연 끝에 replay됨)

uniform sampling은 outdated인 experience를 쓰는 경향이 있었지만 prioritized replay는 본 적 없거나 최근의 experience를 사용(=더 높은 error를 가짐)

CONCLUSION

PER은 학습 속도를 2배를 향상하면서 성능을 높임 → RL 학습을 효율적으로 하는 데에 기여