**GITHUB:** <https://github.com/WannaBeSuperteur/2020/tree/master/DQN>

**[01. Q-learning] [실험: Qlearning.py]**

**Q-learning:** 현재 **상태(s)**에서 **행동(a)**을 취하면 **보상(r)**을 받는다. 이것을 반복하여 **같은/비슷한 상태(s)**에서 **보상값이 최대가 되는 행동**을 찾는다.

**<기본 조건>**

|  |
| --- |
| **s:** 현재 **상태**  **a:** 현재 상태에서 취할 수 있는 **행동**  **s’:** 현재 상태 s에서 **행동 a**를 취하면 도달하는 **상태**  **a’:** **새로운 상태 s’**에서 취할 수 있는 **행동**  **Q(s, a):** 상태 s에서 행동 a를 취했을 때의 **보상값** **(즉시 주어지는 보상 r과는 다름)** |

**<알고리즘>**

|  |  |
| --- | --- |
| 1. 각각의 상태, 행동에 대하여 **보상값을 저장한 테이블(Q-table)을 0으로 초기화**한다.      1. 다음을 충분히(수렴할 때까지) 반복한다.  |  | | --- | | 2-1. 현재 **상태 s**에서 **행동 a**를 실행한다.  2-2. **보상 r**을 받는다.  2-3. **새로운 상태 s’**를 탐색한다. **(s’에서 취할 수 있는 행동: a’)**  2-4. Q-table의 **(s, a)**에 해당하는 칸을 다음과 같이 업데이트한다.   * 즉, **즉시 주어지는 보상값**에 새로운 상태 s’에서 취할 수 있는 **행동 a’에 대한 보상값 중 최댓값의 일정 비율만큼**을 더한다. (**잠재적 보상을 최대화**하는 선택을 하기 위해서 비율을 지정함)   2-5. 현재 상태 s에 새로운 상태 s’를 대입한다. 즉 **새로운 상태에서 탐색을 계속**한다. |  1. 학습이 완료된 후, 상태 s에서 행동을 취하려고 할 때 **보상값 Q(s, a)가 가장 큰 행동 a**를 취한다. |

**<이슈>**

|  |
| --- |
| 보상값 Q(s, a)가 가장 큰 행동만을 실행하면 **다른 행동을 한 이후의 상태를 탐색하지 못하게 하는 문제**가 생긴다. 이를 방지하기 위해 **랜덤하게 행동을 실행할 확률**을 정의하고, 이 확률은 **monotonically decrease하는 함수**가 되게 한다. |

**[02. Q-learning-stochastic] [실험: Qlearning\_stochastic.py]**

**<기본 조건>**

|  |
| --- |
| 기존의 Q-learning과 같이 현재 상태 s에서행동 a를 선택하는데, **이때 행동 a대로 될 확률이 100%가 아니라 원하는 대로 되지 않을 때가 있다**는 것이다. |

**<알고리즘 수정>**

|  |
| --- |
| Q-table의 (s, a)에 해당하는 칸에 **학습률(learning rate)을 적용**하여 업데이트한다. |

알고리즘 수정 결과, Q(s, a)는 지속적으로 학습하면서 얻은 **Q에 대한 approximation, 즉 예상되는 보상 값으로 수렴**한다.

**[03. Deep Q-learning] [실험: Qlearning\_deep.py]**

**Deep Q-learning:** 다음과 같은 Neural Network 구조를 이용하여 학습한다. 즉, **state에 따라 각 action별 reward를 학습**하여 특정 state에서 **가장 reward가 높을 것 같은 action을 선택**한다.

|  |
| --- |
| **State를 이미지로 주고 그 이미지를 CNN으로 딥러닝**시킬 수 있다. Atari 게임을 Deep Q learning으로 학습시키는 상황 등에서 쓰이는 방법. |

원래대로라면 실행 결과는 다음과 같아져야 한다.

|  |
| --- |
| **:** Q table에 따른, **상태** 에서 **행동** 를 취했을 때의 보상의 기댓값 (**prediction**)  **:** 실제로 받는 보상값 (**target**)   * **Prediction과 target의 차이를 줄이는 것**이 목표 |

**<이슈>**

|  |
| --- |
| 다음의 2가지 문제점 때문에 실제로는 **Prediction과 Target의 차이가 발산**한다.   1. **연속된 관측값이 갖는** **상관관계** 2. 실제로 받는 **보상 값이 불규칙**적임 (**noise**) |

**<이슈에 대한 해결 방법>**

|  |
| --- |
| 1. 깊이를 늘린다. (End-to-End learning) |
| 1. 경험을 반복한다. (**연속된 관측값이 갖는** **상관관계** 문제 해결) |
| **의 쌍**을 **replay memory**에 저장한 후, 그것의 minibatch를 replay memory에서 샘플링하여 실행한다.   * 이때 가중치 를 적용하기도 한다. (2015 DeepMind)   + Prediction과 Target의 차이에 약간의 값을 더해서, **Prediction과 Target의 차이가 클수록 뽑힐 확률이 높아지되 모든 샘플에 뽑힐 확률이 약간은 있음을 보장**한다.   + **:** 샘플링 확률이 높은 소수의 샘플만 계속 학습에 사용되는 것을 방지 |
| 1. Double DQN 또는 Dueling Network (**보상 값의 불규칙:noise** 문제 해결) |
| **<Double DQN>**  **가장 보상값이 큰 액션**을 선택하는 network, **각 액션의 값을 추정**하는 network를 서로 분리한다.  **기존** Q Table 업데이트 수식:  **Double DQN**에서 Q Table 업데이트 수식:  **<Dueling Network>**  state stream (**scalar**): : 현재 상태 s가 얼마나 좋은가?  advantage stream (**vector**): : 이 action을 취하는 것이 얼마나 좋은가?  여기서 네트워크 내에서 **두 stream을 합산하면 각각의 특성을 찾을 수 없으므로** 다음과 같이 한다. |

참고문헌:

* <https://jsideas.net/dqn/>
* <http://hunkim.github.io/ml/RL>
* <https://sumniya.tistory.com/19>