Artificial Intelligence: Project 3

Seung-Hoon Na

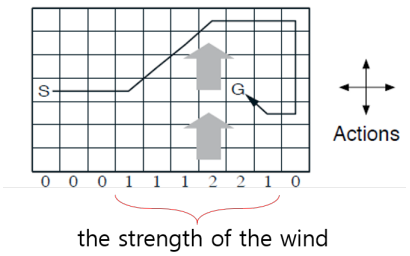
December 12, 2020

1 Sarsa와 Q-learning

1.1 Windy Gridworld

Windy gridworld는 (Sutton 교재 연습문제 6.5) 다음 그림과 같이 8×7 Grid world로, Agent는 `up, down, right, left`의 4가지 action을 수행하면서 시작 상태 S에서 목표 상태 G로 도달하는 것이 목적이다.[1] 추가로, 중앙의 columns들에 바람(wind)이 있어서 이 영역에서는 Agent가 `up`방향으로 해당 바람의 세기(strength)만큼 이동하게 된다. 예를 들어, Agent가 G의 바로 왼쪽에서 `right`를 수행하면 G보다 2칸 더 위에 위치하게 된다. Agent가 G의 바로 오른쪽에서 `left`를 수행하면 G보다 1칸 더 위에 위치하게 된다. (즉, Agent가 이동을 *시작*할 때 위치의 column상 strength만큼 위로 이동한후 해당 action을 수행하는 것과 같다).

Windy gridworld는 Undiscounted espisodic task로 goal에 도달할때마다 reward가 -1씩 주어진다.



1.2 Sarsa와 Q-learning구현

위의 Windy gridworld에 대해서 *epsilon*-greedy action를 이용한 경우 Sarsa와 Q-learning을 구현하시오 (python code). 이때, $\epsilon = 0.1$, $\alpha = 0.5$ 로 한다.

참고로, Sarsa와 Q-learning의 기본 알고리즘은 다음과 같다.

[1] 해당 연습문제에 대한 원문은 다음 Sutton 교재를 참고할 것. `http://incompleteideas.net/book/bookdraft2017nov5.pdf`

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

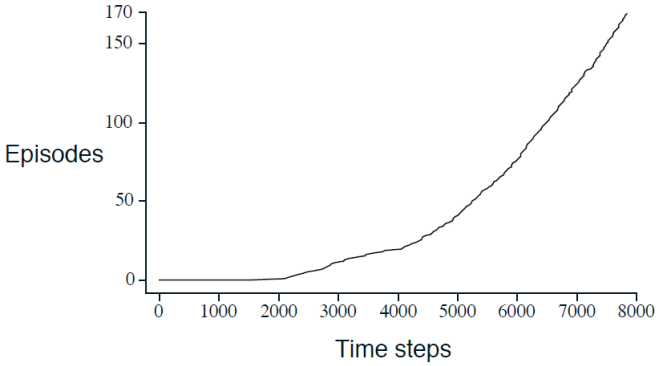
Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

1.3 Sarsa와 Q-learning학습 결과 확인

Sarsa와 Q-learning각각에 대해서 학습한 결과의 Q values와 optimal policy는 별도의 파일로 저장하고 이를 출력하는 python code를 작성하시오.
  출력결과도 report하라.

1.4 Sarsa와 Q-learning 시물레이션

Sarsa와 Q-learning을 시물레이션하여 다음과 같이 time steps별로 수행된 epicodes수에 대한 curves를 비교하시오. (하나의 그래프에서 Sarsa와 Q-learning 곡선이 비교되도록 하면 된다.)



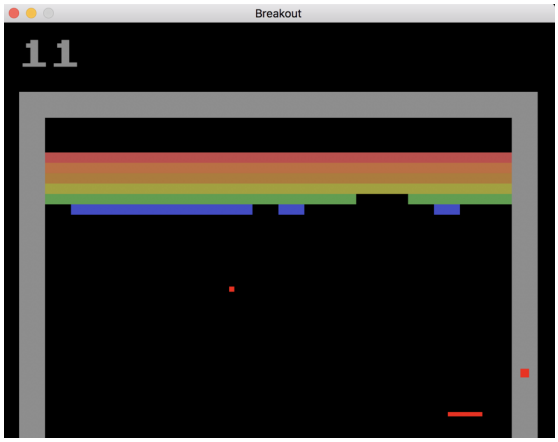
2 Deep reinforcement learning

본 문제에서는 Breakout게임의 action을 학습하기 위해 Convolutional network을 이용한 deep reinforcement learning을 구현한다.

다음은 이를 위해 확장해야 하는 pygame 코드이다.

`https://github.com/aknuck/Atari-Breakout`

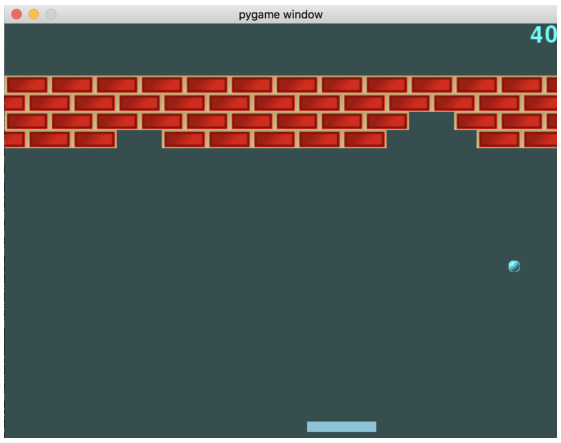
참고로 위의 코드를 설치하여 실행한 게임 화면은 아래와 같다.



또는, 다음의 breakout 코드를 확장해도 된다.

`https://github.com/johncheetham/breakout`

위의 breakout게임 화면은 다음과 같다.



2.1 DQN (Deep Q-network) 구현

DQN은 Experience replay로 (s, a, r, s')로 구성된 replay memory $\mathcal{D}$를 만들고, 이로부터 random mini-batch $\mathcal{D}_i$를 샘플링하여 $\mathcal{D}_i$에 대한 다음 loss function (squared error)를 줄이도록 network의 parameter를 학습하는 것이다.

$$L_i = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i} \left[r + \gamma max_{a'} Q(s', a'; \mathbf{w}_{old}) - Q(s, a, \mathbf{w}) \right]$$

다음은 DeepMind의 DQN 논문에서 기술된 학습 algorithm이다.
(`https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf`)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    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 $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

$Q(s, a, \mathbf{w})$을 위해 convolutional networks으로 구성하여 w학습을 위한 일반적인 DQN 알고리즘을 구현하고 이를 breakout학습에 적용하시오.

단, DQN은 어느 게임에도 적용될 수 있도록 modularity를 유지하도록 하고, Atari breakout code는 state image 및 reward를 추출하기 위해 적절히 수정 및 확장하시오.

다음은 추가 comments이다.

1. pytorch 1.0이상에서 호환되도록 작성하라 (linux ubuntu 플랫폼)

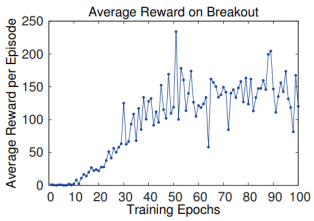
2. convolutional networks의 딥 모델 구조는 3-4개 층 정도로 적절히 구성하면 된다.

3. code실행을 위한 간단한 README 제출할 것

2.2 DQN (Deep Q-network) 적용

DQN을 적용하면서 Training epochs이 진행됨에 따라 Average reward curvs를 그리시오.

다음은 DeepMind 논문의 해당 curves의 예이다.



2.3 DQN (Deep Q-network) 적용: Breakout 시뮬레이션

DQN을 통해 학습된 결과를 저장한 후, 이를 로딩하여 학습된 Agent가 수행한 action에 따라 breakout가 play되도록 시뮬레이션 code를 작성하시오. (실행하면 Agent의 action에 따라 atari breakout가 자동으로 play되어야 함)

2.4 Actor-Critic model 구현

Actor-critic model은 value network $v(S, \mathbf{w})$와 policy network $\pi(S, \theta)$ 두고 value network의 parameter를 갱신할 때는 TD-learning을, policy network의 parameter를 갱신할 때는 policy gradient의 q 함수대신 advantage function $A^{\pi_\theta}(s, a)$를 사용한 다음의 policy gradient를 사용한다.

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla \log \pi(a|S_t, \theta) A^{\pi_\theta}(S, a) \right]$$

Actor-critic model의 학습 algorithm은 다음과 같다 (Sutton 교재 13.5절 참조)

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

$v(S,\mathbf{w})$**및** $Q(s,a,\theta)$**을 위해 convolutional networks으로 구성하여 w,** θ **파라미터을 위한 일반적인 Actor-Critic 알고리즘을 구현하고 이를 breakout 학습에 적용하시오.**

단, 마찬가지로 Actor-critic model은 어느 게임에도 적용될 수 있도록 modularity를 유지하도록 하고, Atari breakout code는 state image 및 reward를 추출하기 위해 적절히 수정 및 확장하시오.

다른 조건은 DQN과 유사하게 설정할 것.

2.5 Actor-Critic model 적용 및 시뮬레이션

마찬가지로, Actor-Critic model을 적용하면서 Training epochs이 진행됨에 따라 Average reward curvs를 그리고, 이를 DQN과 비교하시오.

또한, Actor-Critic model을 통해 학습된 결과를 저장한 후, 이를 로딩하여 학습된 Agent가 수행한 action에 따라 breakout가 play되도록 시뮬레이션 code를 작성하시오. (실행하면 Agent의 action에 따라 breakout가 자동으로 play되어야 함)

2.6 제출 코드 및 결과: 요약

Deep reinforcement learning문제에서 제출해야 하는 코드 및 결과물은 다음과 같다.

1. DQN을 이용한 Breakout 학습기 (학습 결과 저장)
2. DQN을 이용한 Breakout 테스트: performance curve 도출
3. DQN을 이용한 Breakout 테스트: simulation (자동 실행)
4. Actor-critic model을 이용한 Breakout 학습기 (학습 결과 저장)

5. Actor-critic model을 이용한 Breakout 테스트: performance curve 도출
6. Actor-critic model을 이용한 Breakout 테스트: simulation (자동 실행)

3 제출 내용 및 평가 방식

코드는 **python**으로 본 과제 결과물로 필수적으로 제출해야 내용들은 다음과 같다.

- **코드 전체**
- **테스트 결과**: 각 내용별 테스트 코드 및 해당 로그 또는 출력 결과.
- **결과보고서**: 구현 방법을 요약한 보고서.

본 과제의 평가항목 및 배점은 다음과 같다.

- **각 세부내용의 구현 정확성 및 완결성 (80점)**
- **코드의 Readability 및 체계성 (10점)**
- **결과 보고서의 구체성 및 완결성 (10점)**