

과제: Reinforcement learning

[1] 과제 개괄

본 과제는 강화학습에 대한 것이다. 다룰 문제는 강의노트에서 소개한 frozen lake 문제이다. 강화학습 기법 중 Q-learning 기술을 사용하도록 한다. 이 기술을 사용하여 optimal policy 를 구한다 (즉 q 함수를 구한다.) 따라서 이 문제는 control 문제이다. 프로그램은 optimal policy (즉 optimal q-values) 를 학습하는 단계와 그 후에 오는 test 단계로 구성된다. test 단계에서는 구한 q values 를 기반으로 greedy 하게 에피소드를 구해 본다.

본 과제는 두 개의 소과제로 구성되어 두번에 걸쳐 진행된다.

- 소과제-1: table 기반의 Q-learning 방법론을 사용하여 개발한다.
- 소과제-2: function approximation 에 기반한 방법론을 사용한다. 이를 위해 DQN(deep Q-network) 을 사용하도록 한다.

[2] 문제 정의

(2-1) 다음의 환경을 사용한다. 즉 9 X 9 의 frozen lake 이다.

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6									
7									
8									

cell 정의: 녹색: start; 주황색:goal; 흰색: frozen; 회색: hole.

(2-2) actions:

모든 state 에서 다음 4가지 action 이 모두 가능하다: 0(up), 1(right), 2(down), 3(left)

(2-3) rewards:

F/S 로 갈 때 0; G 로 갈 때 9; H 로 갈 때 -9.

[3] 소과제-1 (이미 공지함).

[4] 소과제-2

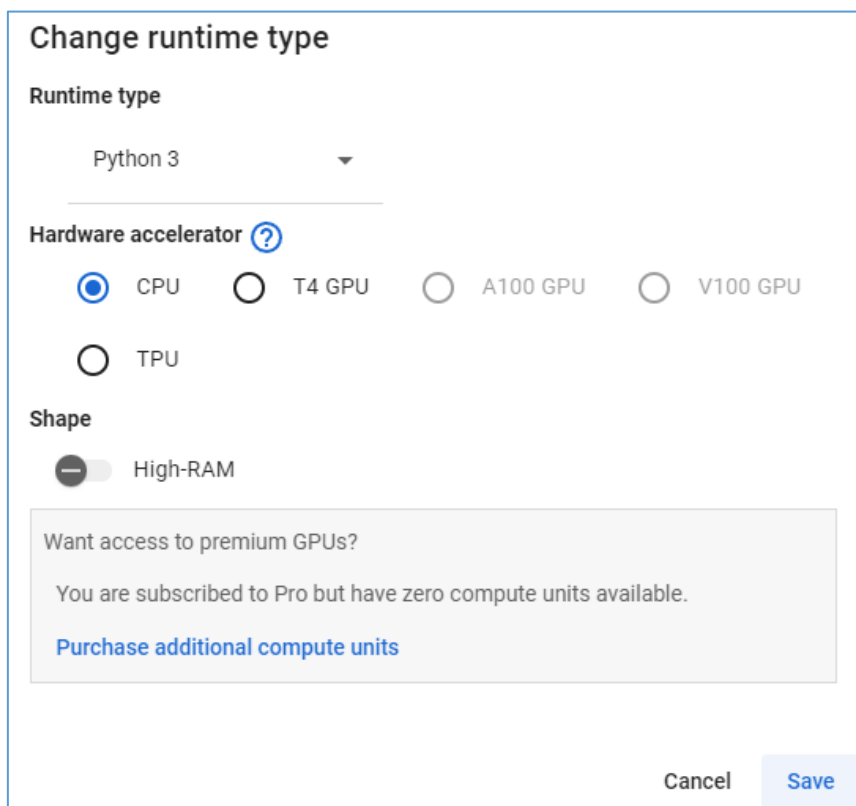
여기에서는 위의 table 기반이 아닌 방식을 사용한다. 즉 table 대신에 function-approximation 기법을 사용한다. function 으로 우리는 neural network 를 사용한다. 특히 유명한 DQN 기법을 사용한다. 프로그램의 절차는 소과제-1 과 동일하게 두 단계 즉 "학습단계"와 "테스트단계"로 구성된다.

학습단계: deep Q-learning 기술을 이용한다. 특히 주요 기술로 correlation을 해결하는 capture and replay 기법과 nonstationary 문제에 대처하기 위해 별도의 두 개의 신경망(separate networks)을 이용하는 기법을 사용한다. 이에 대한 자세한 사항은 강의노트와 guide program 을 참고한다.

• 소과제-2 에서 할 일:

(실험 1) main 실험:

- 강의자료에서 제공한 guide(가이드) 프로그램에서 보여 준대로 "capture and replay(experienced replay)" 기법과 "separate networks 사용" 기법 둘 다를 이용하는 프로그램을 개발한다.
- 학습에서 이용하는 총 에피소드 개수 total_episodes = 100,000 이상 충분히 큰 값으로 한다. 이렇게 많은 데이터를 다루기 위해서는 colab 에서 GPU 를 사용하여 프로그램을 수행하는 것이 좋다.
- 프로그램이 완성 및 준비되어 실행하려 하기 직전에 코랩에서 runtime -> Change runtime type -> 으로 가서 CPU 바로 우측의 T4GPU (또는 그 우측들)를 선택한다. 프로그램 실행 후에 한동안 프로그램을 수행하지 않을 경우에는 다시 CPU 로 되돌려 놓는 것이 좋다.



- 테스트단계에서는 프로그램의 성능을 측정한다. 이를 위해 테스트 단계에서 100 개의 에피소드를 만들도록 한다.
- 생성된 각 episode 의 결과가 정답인지를 체크한다. start state 에서 goal state까지 가장 짧은 경로로 찾아 가야 제대로 정답을 맞춘 것이다.
- 이것은 최단경로의 길이 = 8, total reward = 9 인지를 체크하여 알 수 있다. 100 개의 episode 중 정답을 맞춘 에피소드들이 전체 100개 중 몇 % 인지를 측정하여 보고서에 보고한다.
- (주의: 테스트단계에서 각 에피소드의 move 들은 출력되지 않게 한다. move 들을 모두 출력하면 너무 출력이 많아지기 때문이다).

(실험 2) 학습에 사용되는 episode 개수 (즉 학습데이터 크기) 영향 관찰

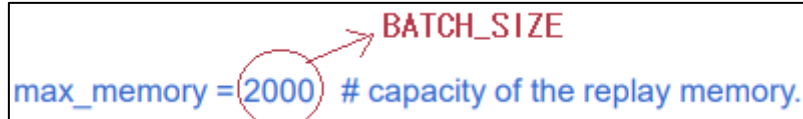
- DQN 에서는 neural network 모델을 사용한다. 모델의 파라미터의 수가 많으면 학습예제의 수가 많아야 한다는 것이 일반적으로 알려진 사실이다.
- 본 탐구에서는 이 점을 실험을 통해 관찰하자. 변수 total_episodes 에 작은 수부터 넣고 실험해 보자. 예를 들면

total_episodes = 5,000 를 넣고 테스트 성능을 관찰한다.

- 이 수를 점차 늘려 가면서(5천개씩) 테스트 성능(accuracy)이 0.5(즉 50%) 이상 나오려면 total_episodes 에 얼마 이상을 넣어야 하는지를 알아 본다.
- 테스트 성능을 보고한다.

(실험 3) experienced replay 기술 효과 관찰

- 이 탐구에서는 experience replay를 사용하지 않는 프로그램의 테스트성능을 측정해 보자.
- 이 실험을 위해 "capture in replay memory" 기능을 제거하기 위해서는 가이드 프로그램을 다음처럼 수정하면 된다.


max_memory = 2000 # capacity of the replay memory.

- 아래에 표시된 것 처럼 수정한다. 빨강 수평선은 해당 줄을 지우라는 의미이다. transition count 를 사용하는 if 문을 변경한다. 그리고 그 내부도 변경한다. 이제는 random sampling 이 필요없다. 그리고 배치 크기 개수의 transition 들이 replay_memory 버퍼에 차면 이들을 그대로 한 배치로 이용하여 학습을 수행한다.

```
# Move to the next state
S = S_

# replay_memory 로 보낸 총 transition 총 개수.
transition_cnt += 1

random_number = random.randint(0, int(BATCH_SIZE/2))

if transition_cnt >= (BATCH_SIZE+3) and transition_cnt % (BATCH_SIZE+random_number) == 0:
##### transition 들을 가져와서 배치 1 개를 만든다. 결과는 batch_transition 에 있다.#####
# replay_memory 에서 꺼내올 위치들을 random 으로 선정하여 random_number 리스트에 넣는다.
if is_replay_memory_full == 1:
#전체 영역에서 가져옴
random_numbers = random.sample(range(0, max_memory), BATCH_SIZE)
else:
# 아직 버퍼가 완전히 차지 않은 상태임. 버퍼의 채워져 있는 부분에서 가져옴.
random_numbers = random.sample(range(0, memory_pos-1), BATCH_SIZE)

# replay_memory 에서 transition들을 꺼내 와서 배치 하나를 batch_transition 에 준비한다.
for i in range(BATCH_SIZE):
    rnum = random_numbers[i]
    batch_transition[i,:] = replay_memory[rnum,:]
##### batch_transition 에 배치준비 완료 #####

is_replay_memory_full = 0 # 다시 0 으로 reset 한다.
```

- 위의 수정이 필요한 이유는 다음과 같다.

. 먼저 replay_memory 버퍼의 크기(max_memory)를 배치 하나의 크기(BATCH_SIZE)로 바꾼다.

. is_replay_memory_full 의 값이 1 이면 배치 하나가 replay memory 버퍼에 준비된 것이다.

이것을 이용하여 배치 하나에 대한 학습을 수행하게 한다.

변수 transition_cnt 는 더 이상 이용하지 않는다.

replay_memory 의 내용을 있는 순서 그대로 batch_transion 에 넣어서 이것을 함수 learning_by_a_batch 가 이용하여 모델을 학습하게 한다.

. 이러한 작업을 한 후 반드시 `is_replay_memory_full` 을 다시 0 으로 돌려 놓는다.

(주의: `memory_pos` 는 앞에서 이미 0 으로 변경하여 놓았음.)

- 실험 (3)에 대해서도 테스트단계에서 100 개의 에피소드 중 몇개가 정답인지를 percent 로 성능을 측정하여 보고서에 보고한다.

- 제출물:

- 프로그램(ipynb파일): 실험 (1), (3) 에 대한 프로그램. (실험 2는 별도의 프로그램 필요없음.)
- 보고서(실험들의 수행창에서 필요 부분들을 캡처하여 넣음).