OpenAI Gym

MountainCar-v0 풀어보기

MountainCar-v0 : Environment

Environment

Observation

Type: Box(2)

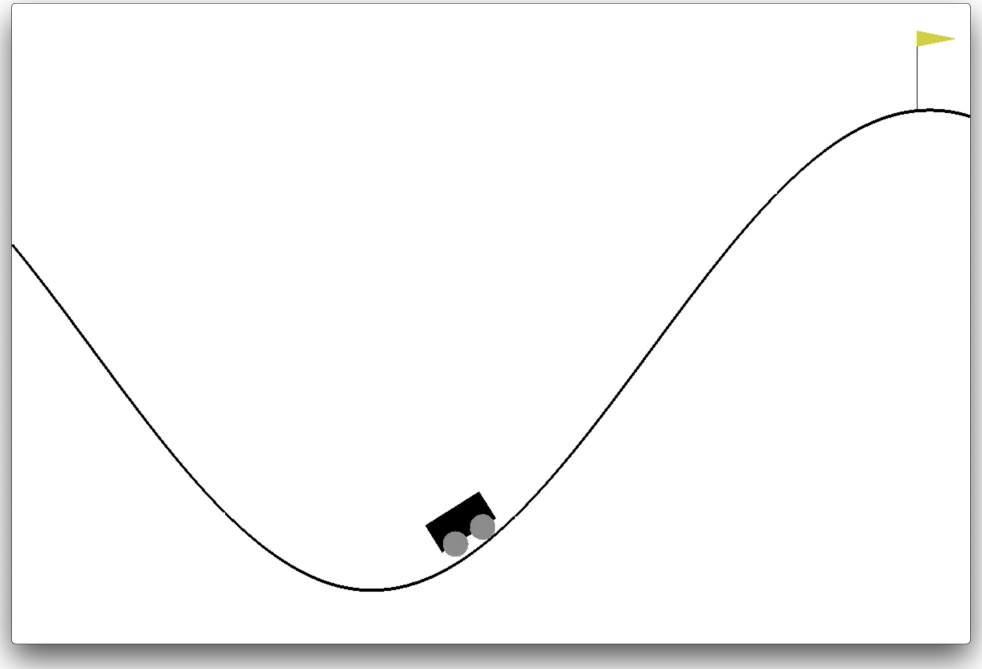
Num	Observation	Min	Max
0	position	-1.2	0.6
1	velocity	-0.07	0.07

Actions

Type: Discrete(3)

Num	Action
0	push left
1	no push
2	push right

MountainCar-v0 : reward



- 목표지점인 0.5에 다다를 때까지 스텝마다 -1씩 계속 받음.
- 200 step이 지나거나 0.5 에 다다르면 에피소드 종료

Cross-Entropy method

- 처음에 그냥 돌려봤는데 random action으로는 상당히 많은 iteration이 지나도 total reward 가 -200 이었음.
- cross-entropy 방법은 elite 에피소드를 학습시키는 방법인데, 성공하지 못한 모든 에피소드는 리워드가 -200이라 개선이 되지 않음.
- reward function을 수정 (observation 값을 보고 임의로 설정하였습니다.)

- `iterate_batches` 함수에서

```
next_obs, reward, is_done, _ = env.step(action)
# 전에 올라갔던 것보다 더 높이 올라가면 reward를 받음.
if abs(next_obs[0] + (math.pi / 6)) > max_position :
    max_position = abs(next_obs[0] + (math.pi / 6))
    if next_obs[0] + (math.pi / 6) > 0:
        reward = max_position * 15
    else:
        reward = max_position * 10

episode_reward += reward
episode_steps.append(EpisodeStep(observation = obs, actio

if is_done:

    if next_obs[0] >= 0.5:
        episode_reward = 10000

    batch.append(Episode(reward = episode_reward, steps =

    max_position = 0.0
    episode_reward = 0.0
```

Cross-Entropy method

- max_position 을 기억하고 그것보다 더 높게 올라갈 때마다 reward를 받게 됨.
- 오른쪽 언덕을 올라갈때의 reward 가중치가 더 높음.
- 목표 지점에 도달하여 에피소드 종료될때 10000의 reward(reward boundary 계산을 쉽게 하려고...)
- 히든 레이어는 1개(128) 사용.
- 100개 배치중에 90% 이상 풀수 있다고 판단했을 때 학습 종료.

로그 화면 및 결과 영상

```
223: loss = 0.363, reward_mean=8688.7, reward_bound=10000.0
224: loss = 0.354, reward_mean=7919.7, reward_bound=10000.0
225: loss = 0.330, reward_mean=7641.4, reward_bound=10000.0
226: loss = 0.336, reward_mean=7071.5, reward_bound=10000.0
227: loss = 0.328, reward_mean=6888.1, reward_bound=10000.0
228: loss = 0.332, reward_mean=8020.8, reward_bound=10000.0
229: loss = 0.330, reward_mean=8118.8, reward_bound=10000.0
230: loss = 0.342, reward_mean=8872.1, reward_bound=10000.0
231: loss = 0.357, reward_mean=8870.8, reward_bound=10000.0
232: loss = 0.338, reward_mean=8586.9, reward_bound=10000.0
233: loss = 0.328, reward_mean=8299.3, reward_bound=10000.0
234: loss = 0.335, reward_mean=8030.2, reward_bound=10000.0
235: loss = 0.345, reward_mean=8595.0, reward_bound=10000.0
236: loss = 0.347, reward_mean=8312.4, reward_bound=10000.0
237: loss = 0.337, reward_mean=9062.7, reward_bound=10000.0
Solved!
```

Deep Q-Network

- Atari 게임을 푼 책의 코드는 convolution layer 를 사용.
- 아타리 게임은 스크린을 observation으로 활용하기 때문에 convolution 방식이 적절하나, MountainCar 는 observation이 2개뿐이어서 부적절하다고 판단.
  (스크린을 읽을까 생각해봤는데 너무 오래걸릴거같아서 포기했습니다. observation 2개로 one-hot encoding 후에 convolution 쓰는 방식도 가능할까요?)
- 일반 뉴럴 네트워크 히든레이어 3개 층으로 시작. (Deep 의 정의는 히든 레이어 2개 이상)

- 사용한 네트워크

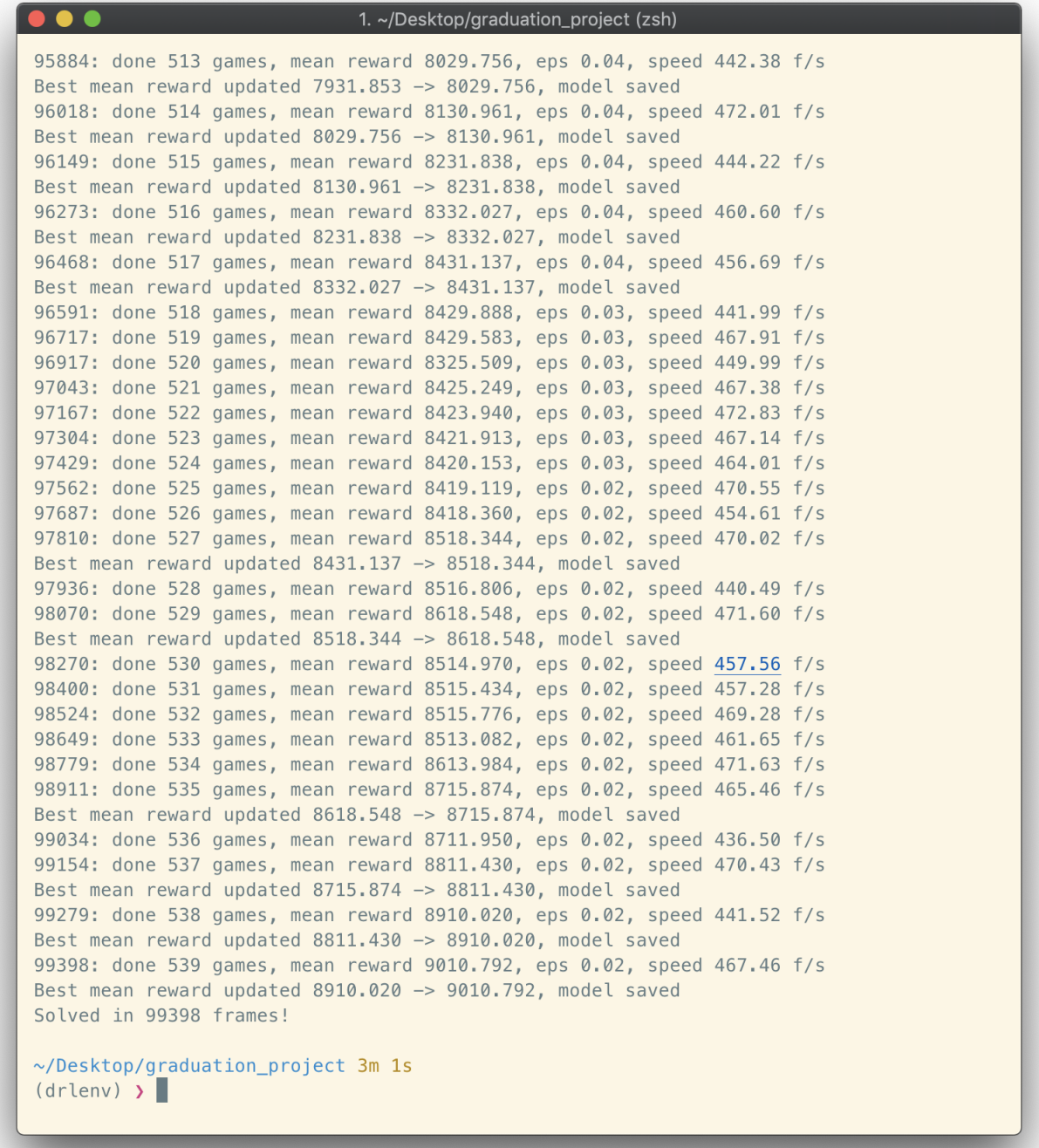
```
class Net(nn.Module):
    def __init__ (self, obs_size, n_actions):
        super(Net, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(obs_size, 128), # 3개의 hidden layer
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            # 3개로 하니까 수렴이 잘 안돼서 2개로 줄임
            # nn.Linear(128, 128),
            # nn.ReLU(),
            nn.Linear(128, n_actions)
        )
    def forward(self, x):
        # 자꾸 더블 텐서 입력한다고 런타임 에러나서 추가
        x = x.float()
        return self.net(x)
```

- `Agent` 클래스의 `play_step` 함수

```
new_state, reward, is_done, _ = self.env.step(action)

# reward rule 수정.
if abs(new_state[0] + (math.pi / 6)) > self.max_position
    self.max_position = abs(new_state[0] + (math.pi / 6))
    if new_state[0] + (math.pi / 6) > 0:
        reward = self.max_position * 15
    else:
        reward = self.max_position * 10

self.total_reward += reward

exp = Experience(self.state, action, reward, is_done, new_
self.exp_buffer.append(exp)
self.state = new_state

if is_done:
    if self.state[0]>= 0.5:
        self.total_reward += 10000
    done_reward = self.total_reward
    self._reset()
return done_reward
```

Deep Q-Network

- cross-entropy와 유사하게 리워드 수정 (수정 안하고 돌렸을 때 개선이 되지 않음..)
- 다른 hyperparameter 는 책의 DQN 코드 그대로 사용.
- 히든레이어 3개 사용했을 때에는 conversion이 어느 선까지만 되고 한계가 있음.
- 히든레이어 2개 사용했을 때에 conversion 속도와 성능이 훨씬 좋았다.
    - input 이 너무 작아서 일까요?
- 최근 100개의 리워드 가운데 90퍼 이상 성공했다 판단했을 때 종료.


1. ~/Desktop/graduation_project (zsh)
95884: done 513 games, mean reward 8029.756, eps 0.04, speed 442.38 f/s
Best mean reward updated 7931.853 -> 8029.756, model saved
96018: done 514 games, mean reward 8130.961, eps 0.04, speed 472.01 f/s
Best mean reward updated 8029.756 -> 8130.961, model saved
96149: done 515 games, mean reward 8231.838, eps 0.04, speed 444.22 f/s
Best mean reward updated 8130.961 -> 8231.838, model saved
96273: done 516 games, mean reward 8332.027, eps 0.04, speed 460.60 f/s
Best mean reward updated 8231.838 -> 8332.027, model saved
96468: done 517 games, mean reward 8431.137, eps 0.04, speed 456.69 f/s
Best mean reward updated 8332.027 -> 8431.137, model saved
96591: done 518 games, mean reward 8429.888, eps 0.03, speed 441.99 f/s
96717: done 519 games, mean reward 8429.583, eps 0.03, speed 467.91 f/s
96917: done 520 games, mean reward 8325.509, eps 0.03, speed 449.99 f/s
97043: done 521 games, mean reward 8425.249, eps 0.03, speed 467.38 f/s
97167: done 522 games, mean reward 8423.940, eps 0.03, speed 472.83 f/s
97304: done 523 games, mean reward 8421.913, eps 0.03, speed 467.14 f/s
97429: done 524 games, mean reward 8420.153, eps 0.03, speed 464.01 f/s
97562: done 525 games, mean reward 8419.119, eps 0.02, speed 470.55 f/s
97687: done 526 games, mean reward 8418.360, eps 0.02, speed 454.61 f/s
97810: done 527 games, mean reward 8518.344, eps 0.02, speed 470.02 f/s
Best mean reward updated 8431.137 -> 8518.344, model saved
97936: done 528 games, mean reward 8516.806, eps 0.02, speed 440.49 f/s
98070: done 529 games, mean reward 8618.548, eps 0.02, speed 471.60 f/s
Best mean reward updated 8518.344 -> 8618.548, model saved
98270: done 530 games, mean reward 8514.970, eps 0.02, speed 457.56 f/s
98400: done 531 games, mean reward 8515.434, eps 0.02, speed 457.28 f/s
98524: done 532 games, mean reward 8515.776, eps 0.02, speed 469.28 f/s
98649: done 533 games, mean reward 8513.082, eps 0.02, speed 461.65 f/s
98779: done 534 games, mean reward 8613.984, eps 0.02, speed 471.63 f/s
98911: done 535 games, mean reward 8715.874, eps 0.02, speed 465.46 f/s
Best mean reward updated 8618.548 -> 8715.874, model saved
99034: done 536 games, mean reward 8711.950, eps 0.02, speed 436.50 f/s
99154: done 537 games, mean reward 8811.430, eps 0.02, speed 470.43 f/s
Best mean reward updated 8715.874 -> 8811.430, model saved
99279: done 538 games, mean reward 8910.020, eps 0.02, speed 441.52 f/s
Best mean reward updated 8811.430 -> 8910.020, model saved
99398: done 539 games, mean reward 9010.792, eps 0.02, speed 467.46 f/s
Best mean reward updated 8910.020 -> 9010.792, model saved
Solved in 99398 frames!
~/Desktop/graduation_project 3m 1s
(drlenv) ❯

결론

- cross-entropy(237 * 100 게임)에 비해서 DQN(539 게임)이 훨씬 적은 에피소드로 agent 를 학습시킬수 있었습니다.
- 그렇지만 DQN 은 매 스텝마다 network를 학습시키고, 리플레이 버퍼에서 샘플링을 해야하며, 일정 횟수(1000스텝)마다 target network와 동기화를 해야합니다. 결과적으로는 비슷한 정도의 시간이 걸렸습니다.
- reward function에 대한 질문: reward function을 어느 정도로 디테일하게 설정해야 할까요? reward function이 자세할수록 학습이 원활해지는 것 같아 보이지만, 또 reward function이 너무 자세하다면 강화학습으로써의 의미가 있을까요?