DQN

taxi-v3

면접 준비로 인해 원 과제를 시간 내에 할 수 없어 교수님께서 공지해주신 대체 과제를 진행하였습니다. 죄송합니다.

train…

```
def train(env, config, qnet):
    optimizer = torch.optim.SGD(qnet.parameters(), lr = 0.001)
    if config.double_dqn:
        qnet2 = copy.deepcopy(qnet)
    else:
        qnet2 = qnet
    criteria = nn.MSELoss()
    num_episodes = config.num_episodes
    for i in tqdm(range(1, num_episodes + 1)):
        train_episode(env, config, qnet, qnet2, optimizer, criteria, episode_count = i)
        if i % config.renew_target == 0:
            if config.double_dqn:
                qnet2 = copy.deepcopy(qnet)
        if i == 1 or i % 50 == 0:
            test(env, config, qnet, i)
    print("Training finished.\n")
def train_episode(env, config, qnet, qnet2, optimizer, criteria, episode_count = -1):
    done = False
    state = env.reset()
    n_steps = 0
    tot_reward = 0
    penalties = 0
    loss = 0
    while not done and n_steps < config.max_time_steps:
        state_t = torch.LongTensor([state])
        epsilon = config.get_epsilon(episode_count)
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample()
        else:
            with torch.no_grad():
                q_hat = qnet(state_t)
                action = torch.argmax(q_hat[0]).item()
        next_state, reward, done, info = env.step(action)
        tot_reward += reward
        new_tuple = (state, action, next_state, reward, done)
        state = next_state
        if reward == -10:
            penalties += 1
        if config.store_and_replay:
            config.insert_record(new_tuple)
            loss_i, step_i = replay(qnet, qnet2, config, optimizer, criteria)
        else:
            loss_i, step_i = replay(qnet, qnet2, config, optimizer, criteria, rec = new_tuple)
        loss += loss_i
        n_steps += 1
```

train 함수는 전체 에피소드의 학습 함수입니다.
train_episode 함수는 에피소드 하나의 학습 함수입니다. train험수 안에서 돌아갑니다.
기존 RL과 다른 점은 학습에 NN을 사용했다는 점과, replay 메모리를 사용한 점, 마지막으로 double network를 사용했다는 점입니다.

기존 RL과 다른 점은 학습에 Q_table대신 NN을 사용한 점과 replay 메모리로 하여금 correlated sample 문제를 해결한 것, 마지막으로 double dqn을 채택하여 non-stationary target문제를 해결했다는 것입니다.

Store and Replay

```
class Config():
  def get_replay_memory(self):
          return self.replay_memory_success + self.replay_memory_fail

      def get_replay_record(self):
          replay_memory = self.get_replay_memory()
          mid_replay = random.randint(0, len(replay_memory) - 1)
          return mid_replay, replay_memory[mid_replay]
def replay(qnet, qnet2, config, optimizer, criteria, num_instances = 5, rec = None):
    loss_i = 0
    for _ in range(num_instances):
        optimizer.zero_grad()
        if config.store_and_replay:
            mid_replay, rec = config.get_replay_record()
        state, action, next_state, reward, done = rec
        if done:
            y_t = torch.Tensor([reward])
        else:
            next_state_r_t = torch.LongTensor([next_state])
            with torch.no_grad():
                q_next = qnet2(next_state_r_t)
                y_t = reward + config.gamma * q_next.max(dim = -1)[0]
        state_r_t = torch.LongTensor([state])
        q_hat = qnet(state_r_t)
        q_hat = q_hat[:, action
        loss = criteria(q_hat, y_t)
        loss.backward()
        optimizer.step()
        loss_i += loss.item()
    return loss_i, num_instances
```

다음 코드가 replay memory에 대한 코드입니다.

Double DQN

```
if config.double_dqn:
        qnet2 = copy.deepcopy(qnet)  # 파라미터 업데이트용 네트워크를 하나 더 만들어줌

...
```

```
if i % config.renew_target == 0:
            if config.double_dqn:
                qnet2 = copy.deepcopy(qnet)  # 네트워크 두개 동기화
```

이번 코드는 Double DQN의 구현에 대한 코드입니다.

저는 Store and Replay와 Double DQN이 없을 때와의 차이를 확인하기 위해 각각의 결과를 확인해 주었습니다.

먼저 SR True, DD True입니다.

1st try

```
Results after 100 episodes:
Average timesteps per episode: 91.79
Average penalty per episode: 0.0
Average reward per episode: -72.47
Training finished.

100%|██████████| 100/100 [00:00<00:00, 175.36it/s]

Results after 100 episodes:
Average timesteps per episode: 111.32
Average penalty per episode: 0.0
Average reward per episode: -92.42
```

2nd try

```
Results after 100 episodes:
Average timesteps per episode: 91.73
Average penalty per episode: 0.0
Average reward per episode: -72.41
Training finished.

100%|██████████| 100/100 [00:00<00:00, 268.27it/s]

Results after 100 episodes:
Average timesteps per episode: 72.25
Average penalty per episode: 0.0
Average reward per episode: -52.51
```

다음으로 SR False, DD False입니다.

1st try

```
Results after 100 episodes:
Average timesteps per episode: 229.64
Average penalty per episode: 0.0
Average reward per episode: -213.26
Training finished.

100%|██████████| 100/100 [00:01<00:00, 63.34it/s]

Results after 100 episodes:
Average timesteps per episode: 298.84
Average penalty per episode: 0.0
Average reward per episode: -283.93
```

2nd try

```
Results after 100 episodes:
Average timesteps per episode: 180.67
Average penalty per episode: 0.0
Average reward per episode: -163.24
Training finished.

100%|██████████| 100/100 [00:00<00:00, 114.28it/s]

Results after 100 episodes:
Average timesteps per episode: 170.75
Average penalty per episode: 0.0
Average reward per episode: -153.11
```

제가 돌려보면서 SR False, DD False인 경우는 수렴이 엄청 느리다는 것을 알 수 있었습니다.

non-stationary targe문제 때문에 수렴이 느린 것으로 알고 있었습니다.

그에 비해 Double DQN을 사용한 모델은 수렴이 그에 반해 빠르다는 것을 알게됐습니다.

그리고 도출된 결과도 많이 차이가 난 것을 볼 수 있었습니다.

시간적으로나 결과적으로나 SR True, DD True 모델이 월등하다는 것을 깨닫게 됐습니다.