

ECS 170: Learning to Play Pong

Ethan He

1 Problem Representation

1. Since we are using a neural network to replace the Q-table, it is easier to use image as state input. As it did in the `QLearner` class:

```
1 self.features = nn.Sequential(  
2     nn.Conv2d(self.input_shape[0], 32, kernel_size=8, stride=4),  
3     nn.ReLU(),  
4     nn.Conv2d(32, 64, kernel_size=4, stride=2),  
5     nn.ReLU(),  
6     nn.Conv2d(64, 64, kernel_size=3, stride=1),  
7     nn.ReLU()  
8 )  
9  
10 self.fc = nn.Sequential(  
11     nn.Linear(self.feature_size(), 512),  
12     nn.ReLU(),  
13     nn.Linear(512, self.num_actions)  
14 )  
15
```

2. The purpose of the neural network in Q-Learning is to replace the lookuo table. In such way, we can train a network for each action, where we use state as input to the network and get \hat{Q} as output.
3. ϵ is the identifier for choosing an action. We generate a random number, if it is greater then ϵ , we do exploitation, so choose the best known action that the Q learner tell us; otherwise, we do exploration, then do random action.
4. See function `act` of `dqn.py`.

2 Making a Q-Learner Learn

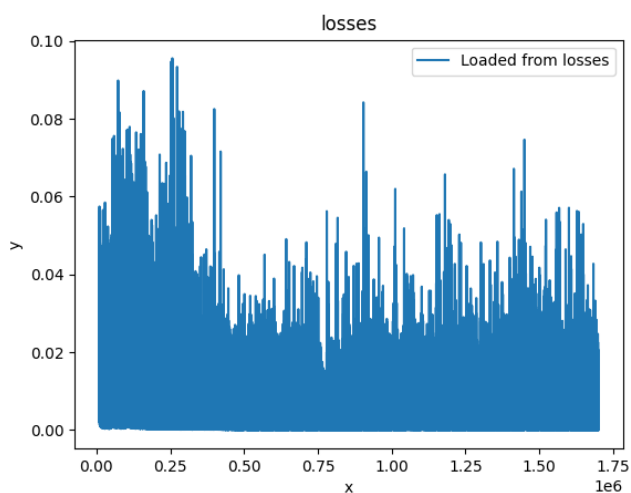
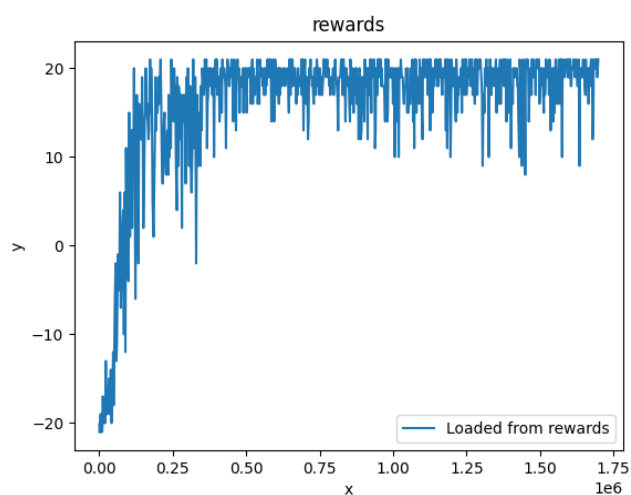
The loss function is the square error formula. In the function, we get the current environment, where `batch_size` controls the sample size we fetch from `replay_buffer` (tensor). And from the random samples, we find the source to calculate the square error. The parameter `gamma` is to control the Q-learner's behavior: as γ get closer to 1, future rewards are given greater emphasis relative to the immediate rewards.

3 Extend the Deep Q-Learner

See program `dqn.py`.

4 Learning to Play Pong

1. See program `run_dqn_pong.py`.
2. See program `run_dqn_pong.py`.
3. Use command `python3 run_dqn_pong.py` to train the model.
4. Plots.



5 Bonus