Assignment 2 - Deep Q-Networks

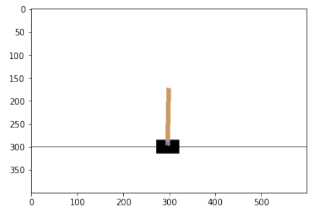
Saeyeon Hwang

April 11, 2022

The date is weird because my labtop recognize time as Korean time

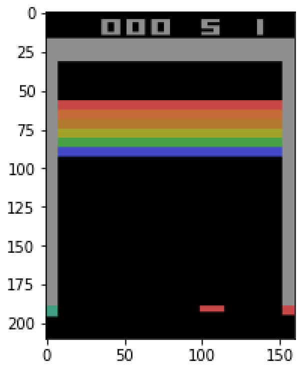
1 Part1 - Exploring OpenAI Gym Environments

1.1 CartPole-v1



There are two possible actions in cartpole problem. Possible actions are moving cart to the left or right. Possible states are defined by 4-dimensional states. They are theta, y, theta dot, y dot. Theta means the angles of the pole, y means the position of the cart, and theta dot, y dot means the each velocities. The goal is to prevent the cartpole from falling over. If cartpole isn't falling over at every timestep, +1 reward is added. The episode ends when the timestep is larger than 500, pole is tilted over 12 degrees from vertical, or the cart moves more than 2.4 from the center.

1.2 Breakout-v0



There are four discrete actions in the Breakout-v0 problem. Do nothing, fire, right, and left. The state is an RGB image of the screen, and it is the array of shape (210,160,3). The goal of the problem is maximizing the score in the Atari 2600 game Breakout. The rewards are provided when bricks are hit.

2 Part2 - Implementing DQN Solving grid-world environment

2.1 Benefits of DQN

1. experience replay By using experience replay in DQN, exploration problem of the agent's would be solved and sample efficiency would be improved. It is because experience replay could make agent remember the past experience to reuse in training. The better replay buffer size would be around 5000 because there is a trade-off between data quality and data correlation.

2. Introducing the target network We implement training with Q-network, so the action would be decided by current state, and they result in next state. By using next state, training Q-network again, and we could get next action. Then using next action, we could get Q value from target network.

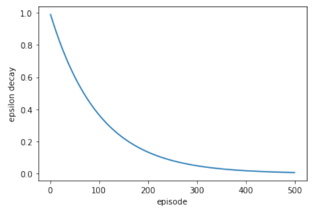
3. Representing the Q function as q(s,w)

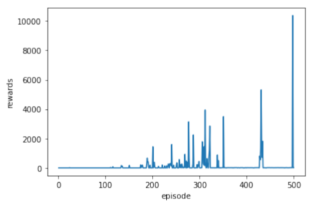
$$Q(s,a;\theta) \rightarrow R + \gamma maxQ(s', argmaxQ(s',a';\theta);\theta^{-})$$

2.2 Grid-world environment

In the grid-world environment, the actions are 4. They are down, up, left, and right. The state is deterministic environment and it is consist of 4x4 states. To figure out training results apparently, the reward values of the environment are changed from assignment1. The coordinates of rewards are [0,3], [1,3], [2,2], [3,2] and they have 2,4,6,8 respective values. And the goal spot has 10 values. When you're on the out of the rewards and goal spot, you'll lose 1 value. The goal is to maximize the rewards.

2.3 DQN implementation

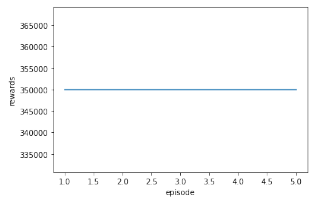




epsilon decay per episode / reward per episode

The above images show the result of training DQN with 500 episodes. Epsilon decays are exponentially and slightly decreased per episode. However, total rewards per episode increases as the model trains DQN, the agent got more rewards as episode proceeds. That means epsilon is decaying well and also the model has trained well.

2.4 Evaluation Results



Run environment for 5 episodes, and let the agent chooses only greedy actions from learn policy. The above plot shows the rewards for 5 episodes. The agent got 349975 rewards for every 5 episodes.

3 Part3 - Improving DQN Solving OpenAI Gym Environments

3.1 Improving vanilla version of DQN

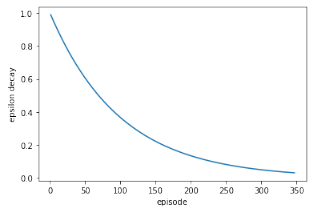
3.1.1 Discuss the algorithm you implemented

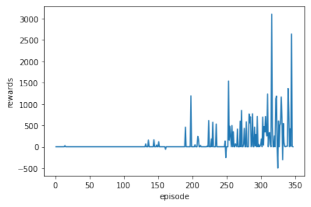
The way to improve vanilla version of DQN is to use Double DQN. Double DQN(DDQN) is the combined model of double q-learning + DQN. Like double q-learning, DDQN make the possibilities lesser that both estimators would overestimate at same action.

3.1.2 What is the main improvement over the vanilla DQN?

The main improvement is the appearance of Q2. The Q1 estimator obtains the best actions, and the Q2 estimator evaluates Q for the above actions. So in the code, the update of target value changed. DQN target function finds the max value of next state. And then multiple with gamma, and adds rewards. On the other hand, DDQN target function finds the max value of next action, and then implement this value on Q-network.

3.1.3 Show and discuss your results after applying your the two algorithms implementation on the environment. Plots should include epsilon decay and the reward per episode.





epsilon decay per episode / reward per episode

The above images show the result of training DDQN with 350 episodes. Epsilon decays are exponentially and slightly decreased per episode. Total rewards per episode increases as the model trains DDQN, the agent got more rewards as episode proceeds. That means epsilon is decaying well and also the model has trained well. By comparing the performance of both algorithms, DDQN is better than DQN

3.1.4 Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

3.2 Applying DQN-based algorithms to solve TWO complex environments