Report of P1 Navigation

In this project, I used DQN with experience replay and fixed Q target, and two extra improvement method: double DQN and Dueling DQN. Each method will be introduced.

DQN:

Traditional Q learning is a very powerful algorithm. However, as the state increases, we cannot store all the states and actions in the q table. In other words, the Q Learning Agent does not have the ability to estimate the value of an invisible state. To solve this problem, we convert q table into a neural network. This is the depth q learning.

DQN uses neural networks to estimate the Q-value function. The input to the network is current and the output is the corresponding Q value for each action.

Experience Replay:

Experience replay transforms the reinforcement learning problem into a supervised learning problem. Since the training samples in a typical RL setup are highly correlated and data inefficient, this will make the network more difficult to converge. One way to solve the problem of sample distribution is to use empirical replay. Basically, store sample conversions and then randomly select from the "conversion pool" to update the knowledge. This is like storing the experience that has been experienced into a memory, and using the experience as data for supervised learning without a certain number of steps.

Fixed Q target:

The target Q network has the same structure as the estimated value. At each certain step, the target network is reset to another. As a result, fluctuations become less severe, leading to more stable training. This is equivalent to removing the correlation between the target network and the network being trained. Make training more stable

Double DQN :

When calculating TD targets, we face a simple question: How do we determine the best action for the next state is the action with the highest Q value?

We know that the accuracy of q values depends on the actions we try and the proximity we explore.

Therefore, at the beginning of the training, we did not have enough information to understand the best course of action. Therefore, using the maximum q value (noise) as the best action may result in false positives. Learning will be complicated if non-optimal actions periodically give a higher Q value than the best best action.

The solution is: When we calculate the Q target, we use two networks to separate the action selection from the target Q value generation. Use our DQN network to choose the best action for the next state (the operation with the highest Q value) and use our target network to calculate the target Q value for the operation in the next state.

Dueling DQN:

We can decompose Q(s, a) into:
•V(s): the value in this state
• A(s, a): The advantage of taking this action in this state (how good this action is compared to all other possible actions in the state).
With DDQN, we want to use two new streams to separate the estimators of these two elements:
• Person who estimates the state value V(s)
• Persons who estimate the merits of each action A(s,a)

Future work:

DQN has many improvement methods, such as prioritized experience replay, reward clipping. I will implement all of them in the future.

And then, In this project, I can directly learn from pixels instead of the vector observation.