Navigation Project

# Project goal

Train agent to collect yellow bananas while avoid blue bananas using unity engine

# Environment

This is large, square environment.

The state space:

37 dimensions (agent velocity, ray-based perception,…)

The action space:

0 - Move forward

1 - Move back ward

2 - Turn left

3 - Turn right

This is episodic task and the goal is to get more than or equal +13 point for 100 consecutive episodes.

# Agent

The agent is implemented base on DQN algorithm. DQN is a Q learning that use TD for estimate value function and use neutral network as function approximator.

Q learning big picture. There are 2 steps: Sample and Learn

0) Initialization

Initialize replay buffer

Initialize parameter for neutral network

1)Sample (Interact with environment)

Choose action A using greedy policy.

Take action and get next state, reward

Store (state, reward, action, next state) into replay buffer

2)Learn:

Collect mini batch from Replay Buffer

Use neutral network to calculate the target value

y\_target = reward + gamma\*max(Q(next\_state,\_)) – Q(state,action)

calculate delta w

update w



# Implementation

QNetwork class

Define NN with 2 hidden layers with size 64, 64

The input size is size of state, output size is size of actions. The values of output is distribution over actions.

There are 2 main functions:

\_\_init\_\_(): create layer and input, output for player. It is a computation graph

forward(): receive input as current state and output is distribution over action. The best action will be the action that have the highest value

Agent class

**Variables:**

Replay buffer: A buffer for store experience and sample mini batch for training

Target network: fix network for calculate target value and will be update

Local network: it is used for training and value of parameters will be updated using gradient decent

**Functions:**

Act()

Call target network to get the distribution over actions and use greedy policy to get the action to act

Step():

Store the (state, action, reward, next state, done) in to reply buffer

Learn():

Using batch sample from replay buffer and use gradient decent to update the parameter

Use target network to calculate the Q-target

Use local network to calculate the expected Q value

Soft\_update()

The value of target network will be update base on Tau parameter, we do not copy 100 local parameters to target

# Result

Training result:



The environment is solved in 369 episodes with average score 13.01