Implementation Procedures

The implementation procedures was carried out as presented in the paper: [Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments](https://arxiv.org/abs/1706.02275)

Q-learning is challenged by an inherent non-stationarity of the environment, while policy gradient suffers from a variance that increases as the number of agents grows. Therefore, an adaptation of actor-critic methods that considers action policies of other agents and is able to successfully learn policies that require complex multi-agent coordination, is needed.

In this work, we propose a general-purpose multi-agent learning algorithm that: (1) leads to learned policies that only use local information (i.e. their own observations) at execution time, (2) does not assume a differentiable model of the environment dynamics or any particular structure on the communication method between agents, and (3) is applicable not only to cooperative interaction but to competitive or mixed interaction involving both physical and communicative behavior. The ability to act in mixed cooperative-competitive environments may be critical for intelligent agents; while competitive training provides a natural curriculum for learning, agents must also exhibit cooperative behavior (e.g. with humans) at execution time.

We adopt the framework of centralized training with decentralized execution, allowing the policies to use extra information to ease training, so long as this information is not used at test time. It is unnatural to do this with Q-learning without making additional assumptions about the structure of the environment, as the Q function generally cannot contain different information at training and test time. Thus, we propose a simple extension of actor-critic policy gradient methods where the critic is augmented with extra information about the policies of other agents, while the actor only has access to local information. After training is completed, only the local actors are used at execution phase, acting in a decentralized manner and equally applicable in cooperative and competitive settings. Since the centralized critic function explicitly uses the decision-making policies of other agents, we additionally show that agents can learn approximate models of other agents online and effectively use them in their own policy learning procedure.

Parameters

BUFFER_SIZE = int(1e5) # replay buffer size

BATCH_SIZE = 250 # minibatch size

GAMMA = 0.99 # discount factor

TAU = 1e-3 # for soft update of target parameters

LR_ACTOR = 1e-4 # learning rate of the actor

LR_CRITIC = 1e-3 # learning rate of the critic

WEIGHT_DECAY = 0 # L2 weight decay

Architecture of the 2 Agents

Actor

1.Two fully connected layers with 200 and 150 units each

2.ReLU applied on a linear layer (200 units)

3. ReLU applied on a linear layer (150 units)

4. Hyperbolic tangent applied on a linear layer (output dim=2)

Critic

1. fully connected layers with 200 and 150 units each

2. ReLU applied on a linear layer (200 units)

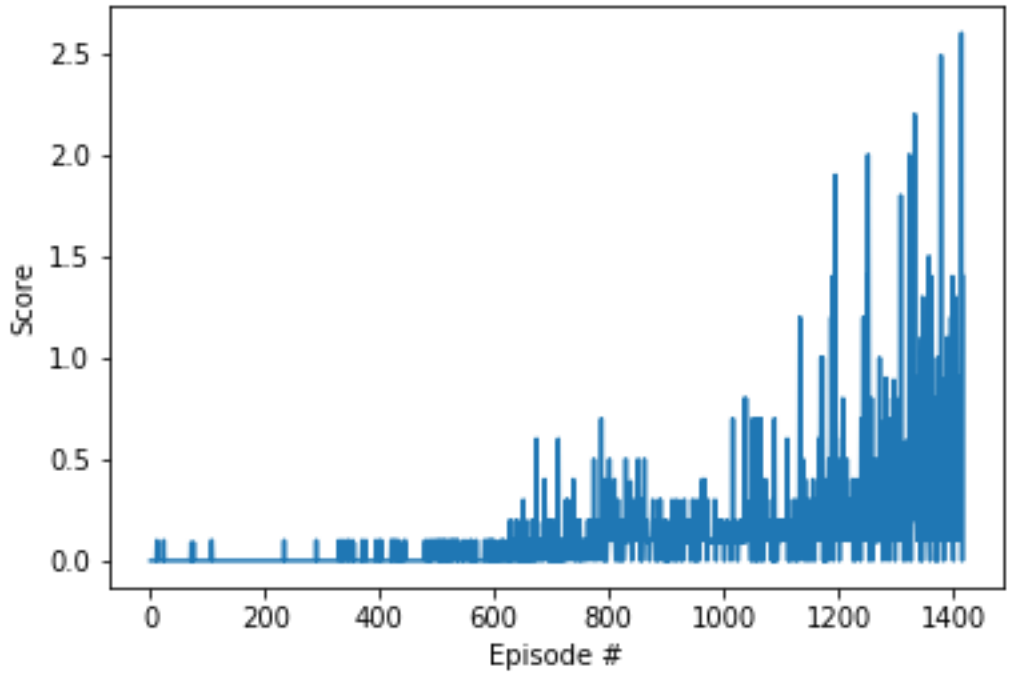
3. Concatenation: add the action at the bottom of the previous output

4. ReLU applied on a linear layer (150 units)

5. Linear layer (output dim=1)

Score and plot of rewards

Episode 100 Average score: 0.004
Episode 200 Average score: 0.002
Episode 300 Average score: 0.002
Episode 400 Average score: 0.010
Episode 500 Average score: 0.018
Episode 600 Average score: 0.036
Episode 700 Average score: 0.077
Episode 800 Average score: 0.119
Episode 900 Average score: 0.155
Episode 1000 Average score: 0.126
Episode 1100 Average score: 0.163
Episode 1200 Average score: 0.242
Episode 1300 Average score: 0.322
Episode 1400 Average score: 0.460
Solved in episode: 1417 Average score: 0.504



Results

The agent solved the task, i.e., reaching an average score of 0.5 over the last 100 episodes, in 1417 episodes. The score of one episode is the best score over both agents.

Further Improvements

The following techniques can be tried out to further improve the performance of the network:

1. Agents can be allowed to operate at different time scales, which improves their ability to use memory and generate consistent action sequent.

2.We can employ the Prioritized Experience Replay method used in reducing training time. This method is used to make training more effective.

3. We can introduce a method to improve the stability of multi-agent policies by training agents with an ensemble of policies, thus requiring robust interaction with a variety of collaborator and competitor policies