Report on solving Tennis Environment

Algorithm

I have implemented Multi-Agent Deep Deterministic Policy Gradient algorithm introduced in [Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments] article. The algorithm description as stated in the article is:

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)\big|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

This algorithm is an extension to multiple agents of the deterministic policy gradient (DDGD) algorithm developed in [Continous Control with Deep Reinforcment Learning Article]. The main novelty is that the critic network input contains actions and observations from all agents.

Implementation

I started with the implementation for the previous project Reacher for DDPG algorithm and extended the implementation to multiple agents. I have made some modifications to replay buffer and extended observation and action space to meet to change the input to critic networks.

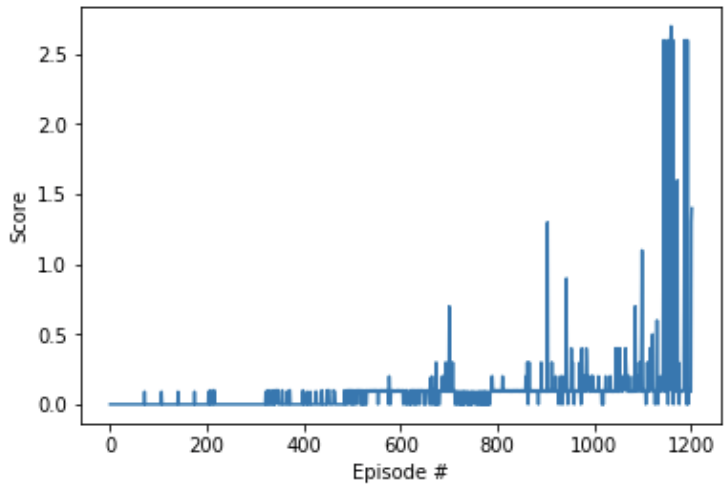
It was difficult to tune the parameters, I have started with networks used for previous projects with three full layers with units of 128 and 164 but there was no convergence at all. I have searched the web for most promising implementations of MADDPG algorithms and investigated implementation from OpenAI to find proposals of good architectures. So I decided to use networks with three fully connected layers and 64 units each for baseline. I have left the batch normalization layer since it stabilised convergence in the last project.

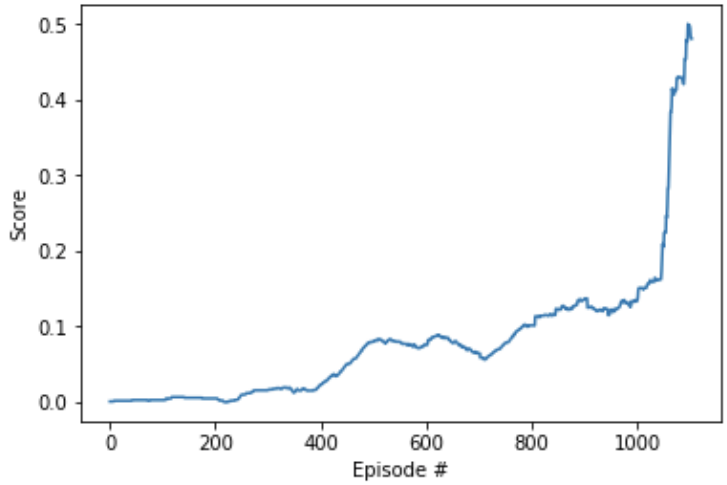
For other hyperparameters I have fixed soft update tau = 0.01 and discount rate gamma = 0.95 and did not change them much. I fixed the batch size to 128 and experimented with learning rates of 0.0001, 0.001 and 0.01. The middle one showed the most promising results. I have tried changing the units to 32 and 128 but did not get any improvements. Then I tried changing the noise sigma parameter since it was crucial for the last project and then I started getting first good results. However, I could not achieve 0.5 score and the whole process was unstable. Then I increased batch size to 256 and the learning rate to 0.003 and got the first results. This is all documented in Tennis.ipynp
Notebook. I also tried experimenting with filling the replay buffer without learning to increase experience space but did not get any better results.

The convergence for learning rate of 0.003 and batch size = 256

I have achieved the objective in around 1200 episodes:

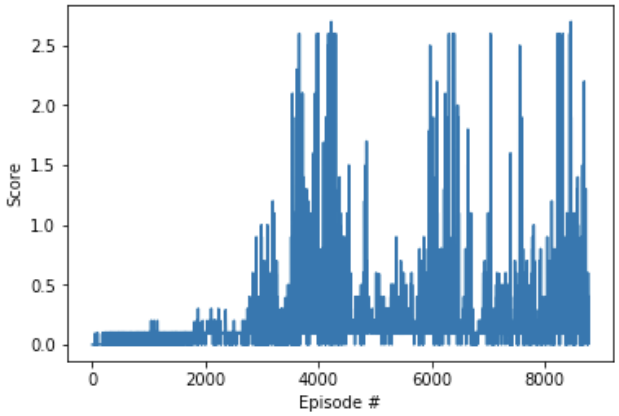
```
Score is over 0.5 so objective is fullfilled. Stopping training.
Training lasted for 20.388347458839416 minutes
```



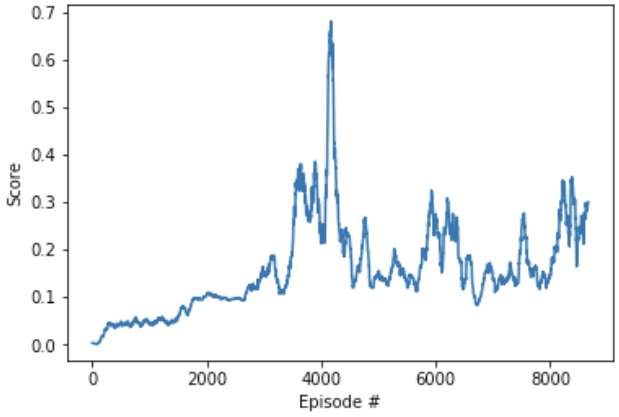


I repeated the experiment with batch size 256 and learning rate 0.004 and achieved the result in 4215 episode I managed tho get a high score of 0.7 here. This is documentedin Tennis2.ipynb

notebook.




```
learning rate 0.001 tau = 0.001 network layers = 64
```



Conclusion and future ideas:

1) The whole training process was highly sensitive to hyperparameters so it was difficult to achieve convergence. The OpenAI implementation uses batch size of 1024 and learning rate of 0.01 with number of episodes as large as 60000. This implies that increasing batch size should stabilise training. For this experiments access to more computational power is needed.
2) The simple DDPG algorithm trained for both agents can produce very good results as noted by many. It seems that it is much easier to achieve convergence in a single agent setting.
3) One could exploit symmetry of the problem to get more use of data.
4) Training first both agents with DDPG and then adding one critic for both agents could prove more easier to get convergence.