Project 3 : Collaboration and Competition

Adithya Subramanian
Deep Reinforcement Learning Nanodegree, Udacity
29/1/2019

Implementation description :

1. SAC.py:
    a. Agent class contains information about the current state representation, allowed actions, The Adam optimizer, Actor network for both the agents, The Critic Networks - local & target for both the agents and replay buffer object is initialized in this class.
    b. The step function of the agent class adds the current state, action, reward, and next state pair into the replay buffer by calling its add function and updates the weights of critic network, actor-network every 4 steps if the size of the buffer is greater than the batch size.
    c. The act function returns the optimal actions as per the current actor network's deterministic policy for both the agent for the current state representation.
    d. The learning function is responsible for computing the mean squared error between the expected Q-value for both the Q-functions and target estimate using V-value. The V-value is learned by using the Bellman optimality equation using the min of the two Q-values.
    e. The Actor is learned using the Critic as a baseline.
    f. The soft_update function updates the target weights using τ parameter and local network weights.
    g. The class ReplayBuffer is the class definition for the buffer memory it is initialized with a deque of pre-defined buffer size which will store a named tuple.
    h. The add function in ReplayBuffer adds the newly visited state transition to the buffer memory.
    i. The sample function samples a complete transition using a uniform sampling technique.

2. Model.py :
    a. The actor model used for transforming the state information vector into the action vector is written in the class Actor.

b. The actor class also returns the log probability of the action sampled and the mean and standard deviation of the action distribution.
c. The Q - critic model used for transforming the state information vector and action vector into the action-value function is written in the class Critic.
d. The V - critic model used for transforming the state information vector into state value function is written in the class Critic.

3. Tennis.ipynb :
    a. The Train function runs multiple episodes of the agent's interactions with the environment for a certain number of transitions. The network stores the average reward collected by the agent over the 100 episodes. Once the agent accumulates over the 0.5+ reward in 100 episodes the environment is considered to solved and model's weight are saved.

Learning algorithm :

Introduction :

Soft Actor-Critic algorithm is derived from entropy based reinforcement learning where we simultaneously try to optimize both the exploitation and entropy. Exploration is proportional to entropy which is a measure of randomness so as consequence the higher the entropy is the more explorative is the agent.

SAC is an off-policy method based policy optimization algorithm. Similar to the TD3 algorithm the SAC also equips clipped double-Q trick to reduce the overestimation of Q - value. The SAC also learns a state-value in order to remove the bias generated during the estimation of critic value from the sampled policy action.

Explanation:

Similar to TD3 simultaneously learn two Q-functions, Q_ϕ_1, and Q_ϕ_2, by mean square Bellman error minimization. The Q- functions are learned by comparing MSE on the same target as shown Equation 1 and the policy is learned by maximizing Q_ϕ_1 similar that of DDPG and TD3 as shown in Equation 2.

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_i}(s,a) - \left(r + \gamma(1-d) V_{\psi_{\mathrm{targ}}}(s') \right) \right)^2 \right]$$

Equation 1.

$$\max_{\theta} \mathop{\mathrm{E}}_{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}} [Q_{\phi_1}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)]$$

Equation 2.

$$L(\psi, \mathcal{D}) = \mathop{\mathrm{E}}_{\substack{s\sim\mathcal{D}\\ \tilde{a}\sim\pi_\theta}} \left[\left(V_\psi(s) - \left(\min_{i=1,2} Q_{\phi_i}(s,\tilde{a}) - \alpha \log \pi_\theta(\tilde{a}|s) \right) \right)^2 \right]$$

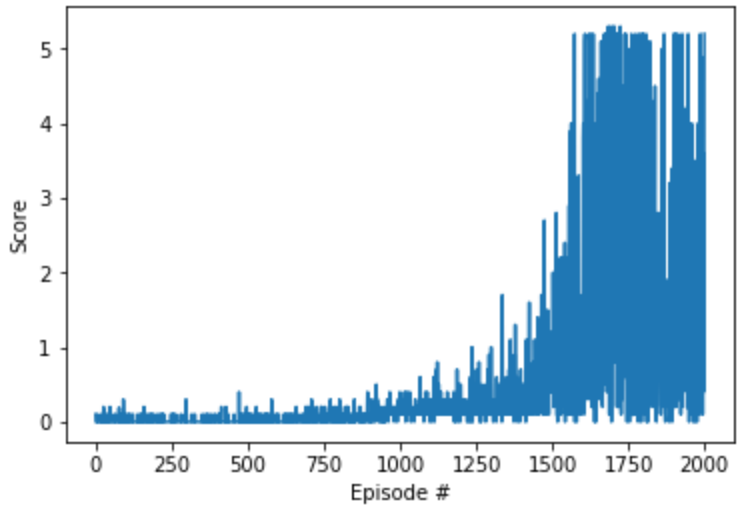
Equation 3.

Unlike the TD3 algorithm the Q - functions are determined using a state-value function. The state value function is used instead of the Q-function because of bias which is present in the actor model which samples action while computing the target value as seen in Equation 1. This might lead to suboptimal results as the actor might always choose an action which is frequently visited might end up exploring less.

Similar to all the DQN variants the state-value function is learned using the target which is the same but it is updated using the soft-update mechanism. This is seen in Equation 3.

As we have two agent hence each action-value network and state-value can observe the entire environment including the action performed by the other agent. The actor can observe its current state information. Hence, we ensure to have centralized training with decentralized execution.

Reward Plot:



Future work:

1. Implementing Actor-Attention-critic to improve the results.
2. Refining the implementation with prioritized replay.

THANK YOU