Project Tennis

Amith Parameshwara

This project aims to create an agent that plays tennis rather that control tennis racket. Multi-agent reinforcement technique as described in the paper (https://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf) is implemented to create the agent.

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

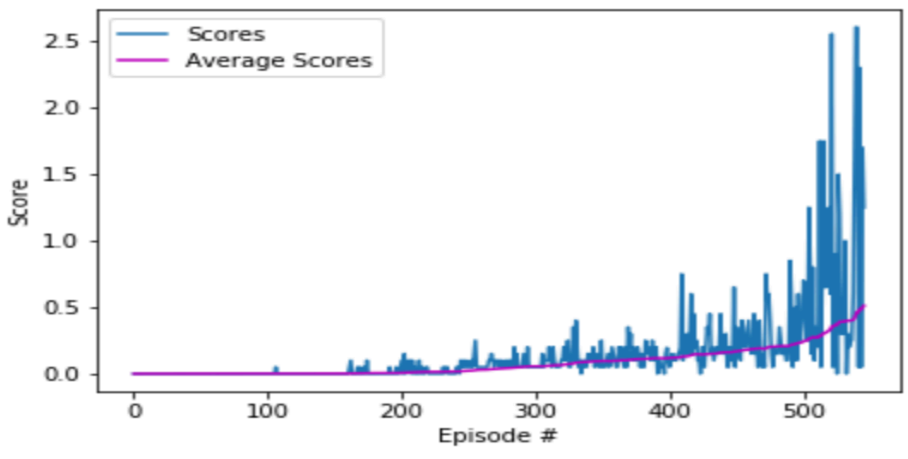
The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.

The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores. This yields a single score for each episode. The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.

Solution

Agent is created using the techniques mentioned in the above paper, which is suitable for multi agent environments. This agent could solve the environment in 446 (average score for 100-episodes in 546th episode) episodes as below.



Algorithm and Hyperparameters

This agent uses 2-layer LSTM network for both actor and critic. LSTM is chosen with the intuition that ideal action of the agent depends on the previous action, state etc and LSTM is good to represent temporal sequences. Number of neurons is set to 256 to learn complex representation and relationship among dimensions of the state.

Learning Method: Both actor and critic are set to learn (i.e. network is updated) every 2 timesteps. Each learning step involves two epochs to stabilize the network.

Learning Rate: Learning rate is chosen to 1e-3 for the actor and 3e-3 after multiple experiments. Further, learn rate scheduler is used wherein the learning rate decays by a factor of 0.01 at each epoch to learn with smaller learning rates and stabilize the network.

Optimizer: RMSPROP is used after trying ADAM initially. RMSPROP is found to be suitable for RNNs in many cases.

Tau (parameter that controls soft update of target networks): Tau is set to 3e-2, slightly higher than the value used for other environments such as continuous control of robotic arm. This is episodic task and also the number of time steps to complete an episode is relatively smaller , therefore smaller Tau value is required to ensure target networks are adequately updated.

Future work

While this agent uses LSTM, ideally LSTM networks should be fed with data with sequence length of more than 1 so that temporal representations is properly utilised. This requires additional pre-processing (minor modifications to replay-buffer) of data to be fed to LSTM. This work is expected to make the LSTM more efficient and help agent solve the problem in fewer episodes.

Other potential improvement is implementation of this paper (https://arxiv.org/pdf/1702.08887.pdf) to improve the experience replay – multi-agent variant of importance sampling to naturally decay the obsolete data.