Report of Project 2 : Continuous Control

Author : Saoussen Chaabnia

Project Description :

For this project we will use Deep Reinforcement Learning Policy-Based-Methods to train an agent (double-jointed arm) to maintain its position at the target location for as many time steps as possible.

In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

Implementation

Actor-Critic method is used to solve this problem. Actor-Critic method is at the intersection between Policy-Based-Methods and Value-Based-Methods. An Actor-Critic uses function approximation to learn a policy and a value function. This achived by using two neural networks , one for the actor and one for the critic. The actor is a neural network which updates the policy and the critic is another neural network which evaluates the policy being learned which is, in turn, used to train the actor.

For our solution, we adopted the **Deep Deterministic Policy Gradient (DDPQ) algorithm** is our implemenation. The actor is used to approximate the optimal policy deterministically, outputting the best action for any given state. The critic learns to evaluate the optimal action-value function by using the actor best believed action.

Some techniques contributed significantly towards stabilizing the training :

Experience Replay: In Experience Replay, we maintain a Replay Buffer of fixed size in which we store some experience tuples as we interact with the environment. After a fixed number of iterations, we sample a few experiences from this replay buffer and use that to calculate the loss and eventually update the parameters. Sampling randomly this way breaks the sequential nature of experiences and stabilizes learning. It also helps us learn from an experience multiple times and recall rare occurrences. Using Experience replay, the value based learning of the reinforcement learning problem is reduced to a supervised learning problem.

Fixed Targets: The idea behind fixed q-targets is to decouples the target from the parameters by fixing the parameters w used to generate the target that we call w-. $w-$ are the weights of a separate target network that are not changed during the learning step. This helps stabilize training. Both the actor and critic are neural networks in which the fixed target is used.

Soft Updates: The target network are updated by mixing the 0.01% of the local network weights with the target network weights that are retrained during each update step.

Hyperparameters:

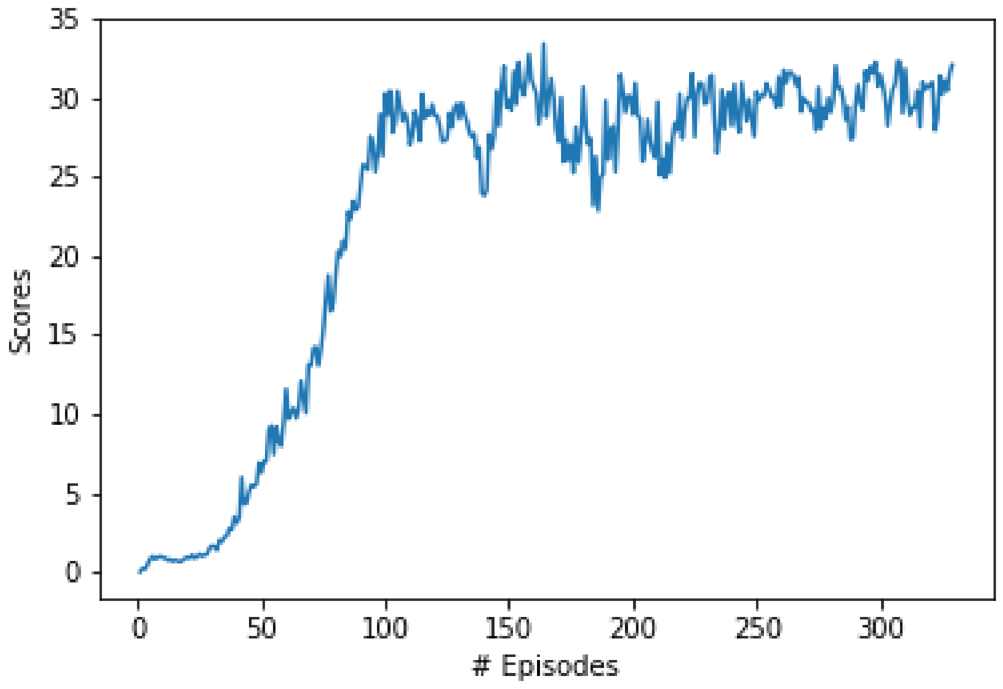
- **n_episodes** : 2000
- **m_tax** : 1000
- **weight_decay** : 0.0001
- **Replay buffer size :**
- **Batch size : 1024**
- **\gamma (discount factor) : 0.99**
- **\tau : 1e-3**
- **Actor Learning rate : 1e-4**
- **Critic Learning rate : 3e-4**
- **Likeness : 0.01**

The Deep Learning Model:

Our actor model constitutes of a two-hidden layers neural network. Layer 1 has 256 hidden units and layer two has 128 hidden units, with Leaky ReLu activation function applied after each fully-connected layer. A tanh function is applied to output layer since the entry action are between -1 and 1. Adam was used as the optimizer for finding the optimal weights.
The critic model has 3 hidden layers of 256, 128 and 128 hidden units respectively. Leaky ReLu is a negative slop of 0.01 is applied to each of them.

Result:

The environment was solved in 229 episodes. The plot of the reward is shown below:



Improvement :

The results can be improved by using the following methods:

- More hyper-parameters Tuning
- Using algorithms like TRPO, PPO, A3C, A2C
- Using the Prioritized Replay Learning Algorithm