In this we’re working with an actor critic neural network. This type of NN is a hybrid system that combines two RL approaches, value-based and policy-based. Actor critic networks have two main components, actor and critic.

* Network Structure:

The aim of actor network is to find some policy that maps each of the states to an action. This network has some fully connected layers that are designed with RELU activation function (Since RELU is a type of activation function that works acceptably well for almost all of the tasks). The final layer of the actor network uses TANH as the activation function.

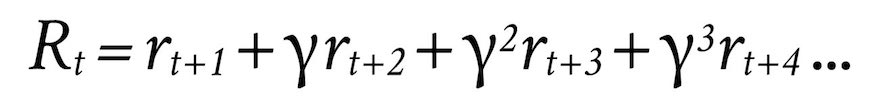
The critic network is used for predicting the value functions for each of the states. This network has two fully connected layers with RELU activation functions followed by the last layer which has a linear activation function that outputs a single value.

* function for computing returns:

The aim of this function is to compute the discounted reward for each time step within an episode in the environment. Actually the discounted return is used to applying the policy that future rewards are less valuable or worth less than the immediate rewards that the agent gets. Discounted reward helps us with having a more accurate estimation of the long-term value of each action.

Gamma is our discount factor that weights the future rewards less.

The discounted reward formula is:



The reason behind using this formula is that in RL learning happens by maximizing the expected rewards.

* Initialization: We first create an instance of the ‘BipedalWalker-v3’ environment. This environment is the one we’re going to use. It has a robot that has to learn how to walk and keep its balance.

Then based on the state and action dimensionality of the environment, we take an instance of the Actor-critic network class and initialize each of the two networks.

Next, the Adam optimizer and the learning rate are set for the training process.

* Train function: For training our agent I’m using Proximal policy optimization algorithm also known as PPO. This algorithm computes gradients and updates the agent’s policy after interaction with the environment. Log probabilities of actions are computed using the policy(actor) network and entropy is being updated during each episode. The advantage is computed using estimated values and returns in each episode. Then two losses one for actor network and one for critic network are computed and combined together to make up the total loss for the agent. Next, the Adam optimizer updates the model parameters. In the end, this function will return the list of rewards.
* Training loop: In this phase, we train our agent for 1000 episodes in total and print the sum of all rewards given to the agent after 100 episodes.
* Evaluation: In this part we evaluate our agent in 10 number of episodes to see the mean of the amount of rewards it got during 10 episodes.

The End