Continues Control – Work Description

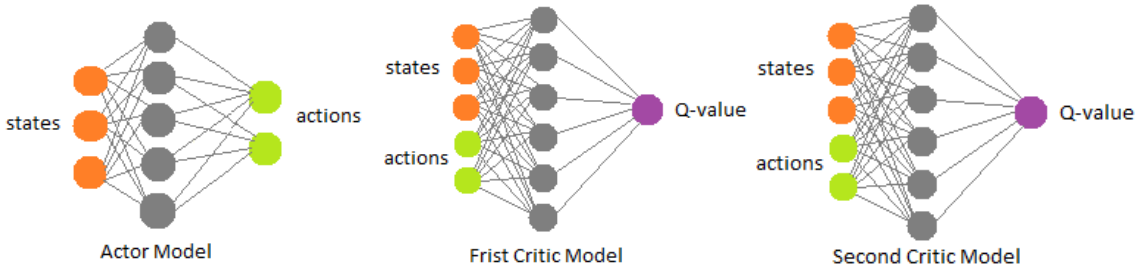
Learning algorithm

I used the special modification of Actor-Critic Deep Deterministic Policy Gradient to Twin Delay Deep Deterministic Policy Gradient method – TD3.

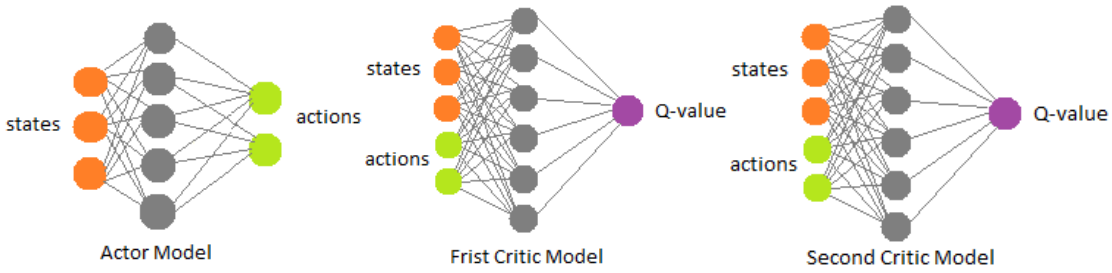
This algorithm considers the interplay between function approximation error in both policy and value updates.

The neural network extension can be described by the next graph representation. Where the extension is for adding second Critic model for local and target model.

Local model:



Target model:



This algorithm deal with problem related to overestimated the bias and high variance because of function approximation error in Deep Q-Learning.

Addressing overestimate bias:

The overestimate bias can be generally address by independent value estimate function. This in actor-critic can be achieved by update local policy during the learning rather than target policy. However, the authors of paper works found out, that by slow-changing the policy from local to target they become to similar to be independent. So instead, they use the pair of critic model. This results to two Q-value for same state “s”. The overestimated is than simply reduce by taken the minimum which ensures to bound the less biased value estimate.

Addressing high variance:

The high variance provides the high noise in gradient update. The high variance is the problem where the value function estimate is built from temporal difference error made from subsequent states. By TD

equitation $Q(s, a) = r + \gamma E[Q(s_0 , a_0)] - \delta(s, a)$ the value estimate is actually the estimation of future reward and TD estimated error. This means that the variance of estimate value is actually the sum of variance of future reward and estimation error. By given the high discount factor the variance grows. In actor-critic algorithm the approach to reduce this updates error variance is to build the target network as this is well known for increase the stability in deep reinforcement learning. The authors of work paper validate this by testing different Polyak update params as well as direct learning policy.

Both of this extension highly increase the stability of deep learning process. The authors documented they work and confirm this by couple of tests. The full paper work can be seen at https://arxiv.org/pdf/1802.09477.pdf

My extension - Local exploration:

My extension to continues action-state environment. How-ever I found one main disadvantage during the learning process. This disadvantage is highly related to environment (continues action-state environment) and process of discovering the interaction by agent. The basic idea of how the agent learns, is to perform the random actions in first places. From random action the agent received the reward. The high disadvantage of this approach is to learn just by random action. The agent is not able to find enough "good" actions in reasonable amount of interaction. This is usually satisfy/improve by "sharing knowledge" between multiple agent. Where multiple agents interact with independent (but same environment) and they share their knowledge. But I think either this approach has still the main disadvantage. And It is just random actions. Just by intuition the people do not learn in this way – they do not perform just random action. On the other hand, when the people get the reward or punishment, they start to explore the local situation. The local explore approach is possible in this environment because the action space is continuing and the reward is immediate. This can be described by: when state is in "reward/close to reward" each next action brings the environment into "closer / far to reward". But by performing the fully random action, this information is simply lost. Do not to loss this information, the next action must be "closer" to previous action. And not just fully random. I improve the random exploration phase by adding "local exploration". During the random learning phase (parameter local_explore_unti_steps) when agent receive reward, he starts the local exploration by change the action just by very small randomness (parameter local_explore_step). This approach increases the number of "good states" and speed up required learning steps.

Hyper params:

warm_up = 5000 – the number of fully random steps, without any learning process

memory_batch_size = 100 – the number of random mini-batch smaples

learn_every = 5 – how often to perform the cirtic model learing warm up.

policy_freg_update = 3 – how often to use the polyak update.

polyak_tau = 0.008 – the size of update between local and target policy

discount = 0.99 – discount factor

local_explore_step = 0.005 – the amount of randomness in local exploration learning phase

local_explore_unti_steps = 40000 – the number of steps until which the agent perform local exploration

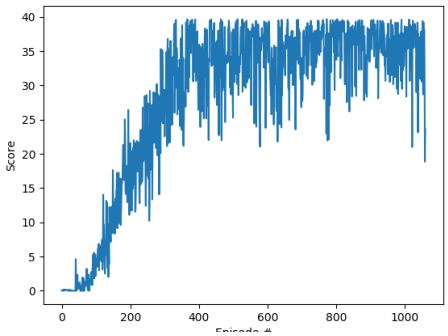
Actor NN:

Two hidden layers with 400, 300 neurons, with tanh output action function.

Critic NN:

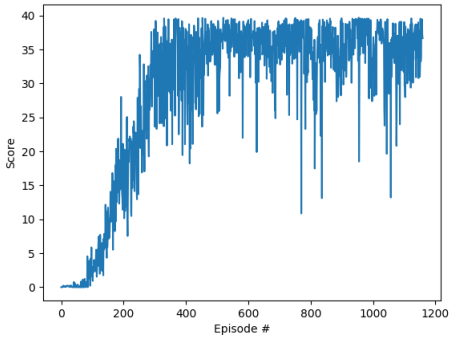
Two hidden layers with 400, 300 neurons and RELU output activation function.

As the algorithm of Twin Delayed Deep Deterministic policy gradient algorithm is describe in paper work, so my work was focused on my extension of local-exploration.



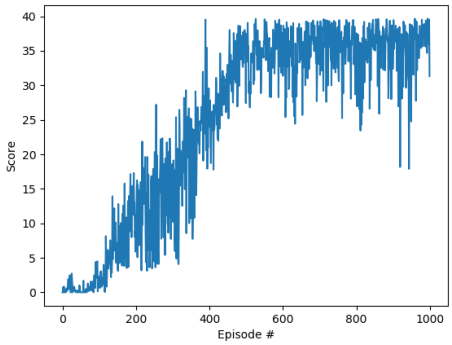
local_explore_step = 0.005

local_explore_unti_steps = 40000



local_explore_step = 0.001

local_explore_unti_steps = 40000



No local exploration

The explore step must be choosing carefully, because it is very dependent on current environment. I try either higher explore steps like 0.1 / 0.05, but they had not such an improvement. Simply the environment does not react “good” for this high randomness. On other hand it does shows that choosing the very small explore steps can lead into over-estimate the local maximum and later into higher variance in future.

I approve the local exploration have a improvement to how fast the agent learns.

Ideas of future work:

How-ever the algorithm of local exploration is very simple it shows the “good direction” of improvement. The disadvantage of this approach is actually to high dependence on environment. Incorrect setup that can lead into higher variance than the simple TD3 algorithm. The improvement can be done in slightly different approach to local-exploration. The exploration can be set during the whole process and not just during random “warm up”. Either the failed results when agent has significant knowledge can be used to explore the local “wrong exploration” in similar approach used before. The challenge here is to set-up start and end of exploration condition. The “good exploration” can have a simple end condition: when reward is zero stops explore. The “wrong exploration” can starts when reward is not enough compared to current agent knowledge. The exploration can be done in similar why – small action steps, but the start of exploration condition must be defined dynamically based on different between current reward and mean of for example last 100 reward. And either end of exploration condition must be defined not just stops local explore when improving the reward. But either in way: if agent is not able to learn from local “wrong exploration” (means the reward does not increase) do not learn at all from that situation. The reason is that for example: if agent tries to move ball, and the ball falls down because of wrong action – the ball is on the floor. The agent do not fixed this situation just by “locally moving hand“, he must bend down and hold the ball, which is completely different action.