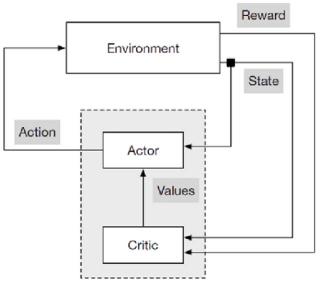
Solving Continuous Action Environment from OpenAI gym

Maria Grazia Berni

May 20, 2022

Actor Critic Configuration

- Q Learning and Deep Q Learning methods can't handle continuous action spaces
- However, innovations from Q Learning, like Replay Memory and target networks, can be applied to actor-critic methods.

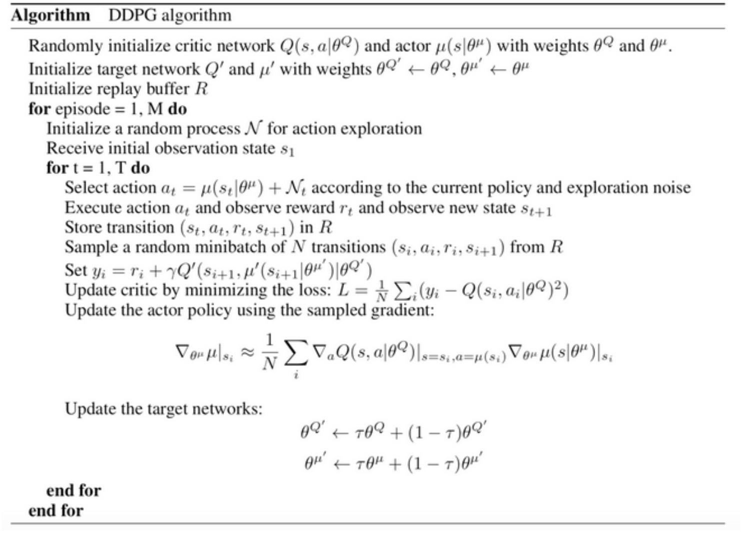


Actor Critic

DDPG

- Deep Deterministic Policy Gradient is an actor-critic based method.
- It uses, together with the main actor and critic networks, one target actor and one target critic network to reduce the per-update errors.
- The target networks are updated performing a soft copy every time step
- the exploratory behavior of the algorithm is guaranteed adding noise to the action selected by the actor
- It's an off policy method

DDPG Algorithm



Algorithm DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
   Initialize a random process $\mathcal{N}$ for action exploration
   Receive initial observation state $s_1$
   for t = 1, T do
      Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
      Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
      Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
      Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
      Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
      Update the actor policy using the sampled gradient:
            $$\nabla_{\theta^\mu} \mu|_{s_i} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$
      Update the target networks:
            $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
            $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
   end for
end for
```

Overestimation Errors

- Overestimation bias is present in actor-critic methods such as DDPG.
- In Q learning we get this bias performing the maximum over action, but there is no max-update rule in actor-critic methods, so where does this bias come from?
- Even if this maximization bias is not so explicit as in Q learning, there is some implicit drive to maximize scores over time using stochastic gradient descent, and in a deep reinforcement learning setting, which typically has high variance, this process can lead to incorrect estimations.

TD3

In addition to the maximization bias problems, DDPG can be quite sensitive to the hyper-parameters involved, and spikes in the action value estimates (which are potentially erroneous) are exploited by the policy network and derails the learning. The TD3 algorithm addresses the function approximation errors in DDPG.

- To address overestimation bias, TD3 uses two target critics independently trained and a Clipped Double Q-Learning algorithm, so the target value is computed as:

  $$y_1 = r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi_1}(s'))$$

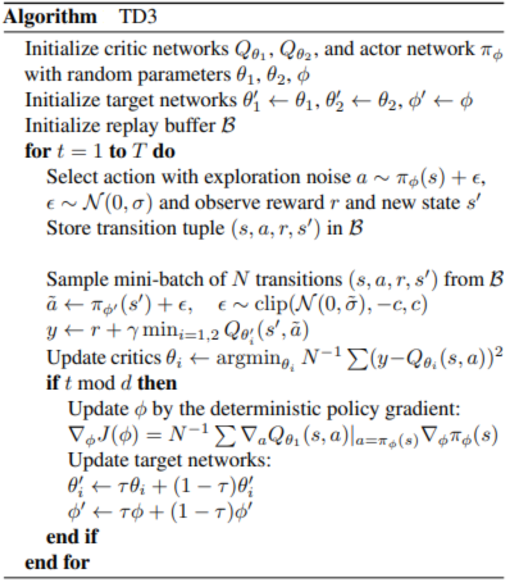
  This update rule may induce an underestimation bias, but this is not a problem, because underestimated actions will not be explicitly propagated through the policy update.

TD3

- To minimize the per-update residual error which, given the nature of td-learning, tends to accumulate, it uses target networks and the policy network is updated at a lower frequency then the value networks, in order to first minimizing the error before introducing a policy update.
- Reduce the variance of the target through policy smoothing regularization, adding clipped noise to the next state actions:

$$y = r + \gamma Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon)$$
$$\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma), -c, c)$$

TD3 Algorithm



Algorithm TD3

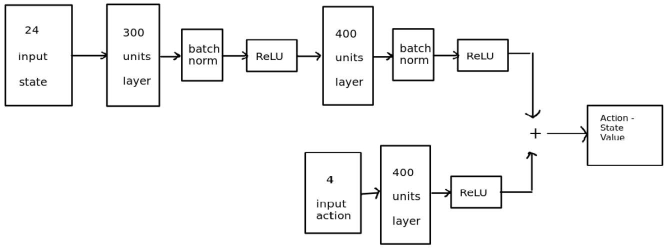
```
Initialize critic networks $Q_{\theta_1}$, $Q_{\theta_2}$, and actor network $\pi_\phi$
with random parameters $\theta_1$, $\theta_2$, $\phi$
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ to $T$ do
    Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$,
    $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward $r$ and new state $s'$
    Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$

    Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
    if $t$ mod $d$ then
        Update $\phi$ by the deterministic policy gradient:
        $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
        Update target networks:
        $\theta'_i \leftarrow \tau \theta_i + (1 - \tau)\theta'_i$
        $\phi' \leftarrow \tau \phi + (1 - \tau)\phi'$
    end if
end for
```

Environments

The DDPG and TD3 algrithm have been tested respectively in the environments BipedalWalker-v3 and BipedalWalkerHardcore-v3. The goal of both environments is to train a robot to run without falling. In the Hardcore version there are some obstacles in the environment. 300+ points of reward are given if the robot runs to the end, if the robot falls it gets -300. The state space is made of 24 continuous observation, consisting in hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements. The continuous action space consists of the torque on both the knee and hips. The environment is considered solved when, training for at most 1600 episodes in the base version and 2000 in the hardcore version, the total reward is greater or equal then 300.

DDPG Network and Hyperparameter

The paper "Continuous control with deep reinforcement learning" suggests to use as Critic a low-dimensional network with 2 hidden layers of 400 and 300 units, including actions in the 2*nd* hidden layer, concatenating it with the input of the layer. During experimentation I found more performing to use the following architecture for the Critic:

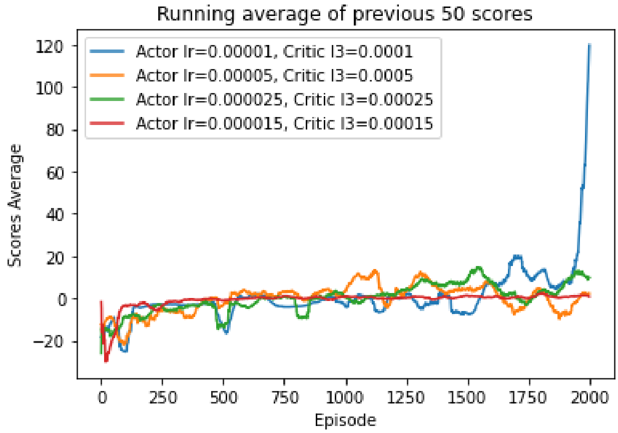


DDPG Network and Hyperparameters

As actor I used the same configuration in the paper, but adding a batch normalization layer before each ReLU. Critic and Actor use the Adam optmizer, and the critic also use L_2 weight decay of 10^{-2} as regularization. Also the soft target update has the value of $\tau = 0.001$, us suggested in the paper, and the batch size is equal to 64. The paper suggest a learning rate of 0.001 for the critic, and 0.0001 for the actor. For the exploration noise both Ornstein-Uhlenbeck process and a norrmal decreasing noise have been used.

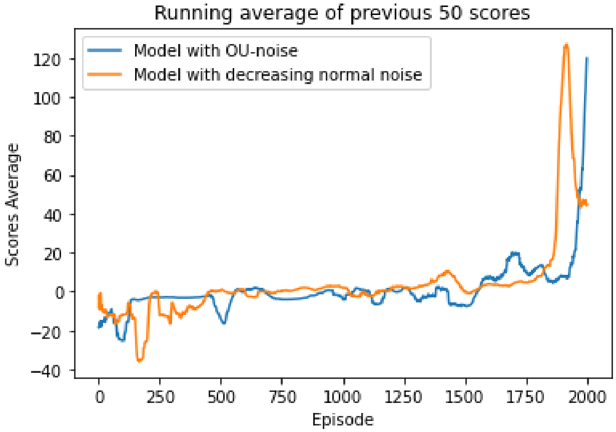
DDPG Training

Running average reward during the training of the agent with different values for the learning rate and using OU-noise.



DDPG Training

Running average reward during the training of the agent with actor $lr = 0.00001$ and critic $lr = 0.0001$, with OU-noise and constant normal decreasing noise.



Considerations

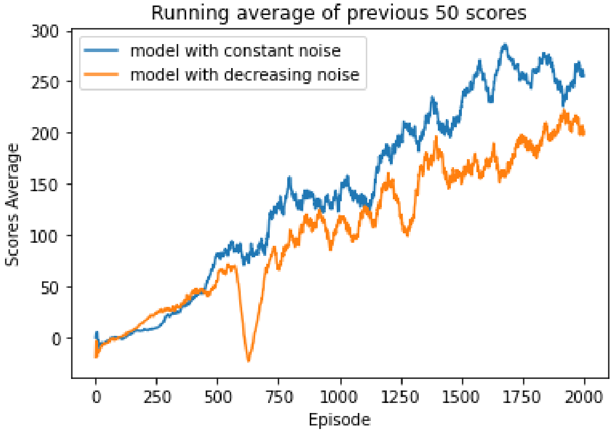
The DDPG algorithm is not able to solve this environment. In any case, with the critic learning rate of 0.0001 and actor learning rate of 0.00001 it reach the better performance, which widely degrades even changing slowly the value of the learning rates, and this emphasizes how the algorithm is sensitive respect to the hyper-parameter selection.

TD3 Network and Hyperparameters

The Critic and Actor network structure has been kept fixed during the experimentation. It consists of two hidden layers of 400 and 300 neurons. In the case of the Critic the action are given as input in the first layer, together with the state. The networks use the Adam optimizer. Rectified Linear Units are used as activations, except from the output of the actor, which uses a hyperbolic tangent, while the output of the critic doesn't use any activation. The TD3 agent has been trained using constant normal noise, with scale 0.1, and using decreasing normal noise, from a scale of 0.25 and decreasing to the value of 0.02. The algorithm has been tested with various learning rate for the actor and critic networks. The paper suggest to use a critic and actor learning rate equal to 0.001

TD3 Training

Running average reward during the training of the agent with paper learning rate values with constant normal noise and decreasing normal noise for exploration.



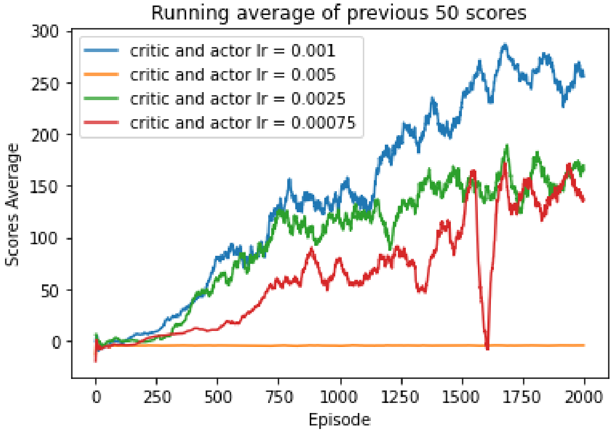
TD3 Testing

The model with constant noise performed better during the training. These two model have been tested for 200 episodes, without any noise in the actions, obtaining the following results:

- Model with constant noise : average reward $= 203.18$, maximum reward $= 311.77$, minimum reward $= -81.23$.
- Model with decreasing noise : average reward $= 181.22$, maximum reward $= 305.6$, minimum reward $= -74.81$.

TD3 Training

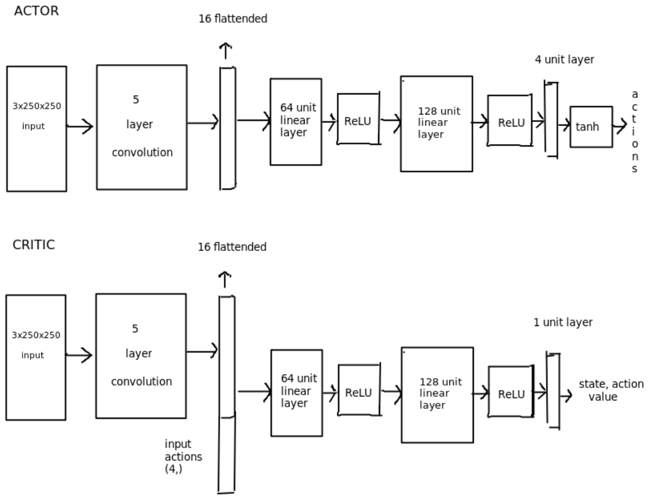
Running average reward during the training of the agent using different learning rate values and constant normal noise for exploration.



Using TD3 with pixel information

One of the goal of reinforcement learning is to reach control with pixel information. To do that, I have modified the interactions with the environment to obtain an image as state, then, using the proper wrapper classes, I cropped and resized it and stack 3 of them together (so that the agent can estimate accelerations). Finally the input state has dimension $3x250x250$. The actor and critic use 5 layers of convolutions, each one with 16 channels, kernel size $5x5$ and stride 2, only the last convolutional layer has stride 1. After each layer I applied batch normalization, a max pool layer with kernel size $5x5$ and stride 2 (after each layer except the first) and ReLU activation.

Actor and Critic Architecture



Training

Conclusions

The TD3 algorithm is able to solve the BipedalWalkerHardcore-v3 environment with the hyper-parameters suggested in the paper "Addressing Function Approximation Error in Actor-Critic Methods" and is quite robust to the change of the hyper parameters, in fact, even if it's no more able to colve completely the environment, the performance does'n widely decay as in the case of the DDPG algorithm. The same is true for the other hyper parameters.

References

Scott Fujimoto, Herke van Hoof, David Meger (2018) *The Book*, Addison-Wesley Professional.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra (2015) *Continuous Control with deep Reinforcement Learning*

Richard Sutton, Andrew Barto (2018) *Reinforcement Learning: an introduction*

Enes Bilgin (2020) *Mastering Reinforcement Learning with Python*