1. Learning Algorithm:

The source code and weights of a trained agent are placed in the main directory. The algorithm is implemented following these guidelines:

- The reinforcement learning algorithm is the DDPG (Deep Deterministic Policy Gradient).

  DDPG belongs to the category of Actor-Critic algorithms. The Actor directly maps states to actions through a policy while the critic performs estimates through a value-based method.

  Both actor and critic use a local and target networks. The reason is the networks in use are not constantly updated, thus the learning is more stable.

  In this case the local and the target network are being hard updated following the parameter indicating the number of steps. The actor network is updated every 4 timesteps during 4 timesteps and the following 4 timesteps it runs without update. The critic network is updated every 10 timesteps during 10 timesteps and the following 10 timesteps it runs without update.

  The "soft update" function with TAU hyperparameter = 1 is used to do the hard update.

- Neural Network configuration:

  Actor Network (x2):
  - Hidden layers are composed of -> 256 -> ReLU -> action_size -> tanh
  - The optimizer is ADAM and
  - The learning rate = 1e-4

  Critic Network(x2):
  - Hidden layers are composed of -> 256 -> LeakyReLU -> 256 -> LeakyReLU -> 128 -> LeakyReLU -> 1
  - The optimizer is ADAM and
  - The learning rate = 3e-4

- Other algorithm configurations:

o BATCH_SIZE = 1024	# minibatch size to be processed by the algorithm
o GAMMA = 0.99	# discount factor for future expected rewards
o update_steps_actor=4	# update cycle for the actor/critic network (x in a row,
o update_steps_critic=10	without update, x in a row with update)
o TAU = 1	# for soft update of target parameters TAU is the rate at which this update takes place so that the network weights do not change suddenly but smoothly.

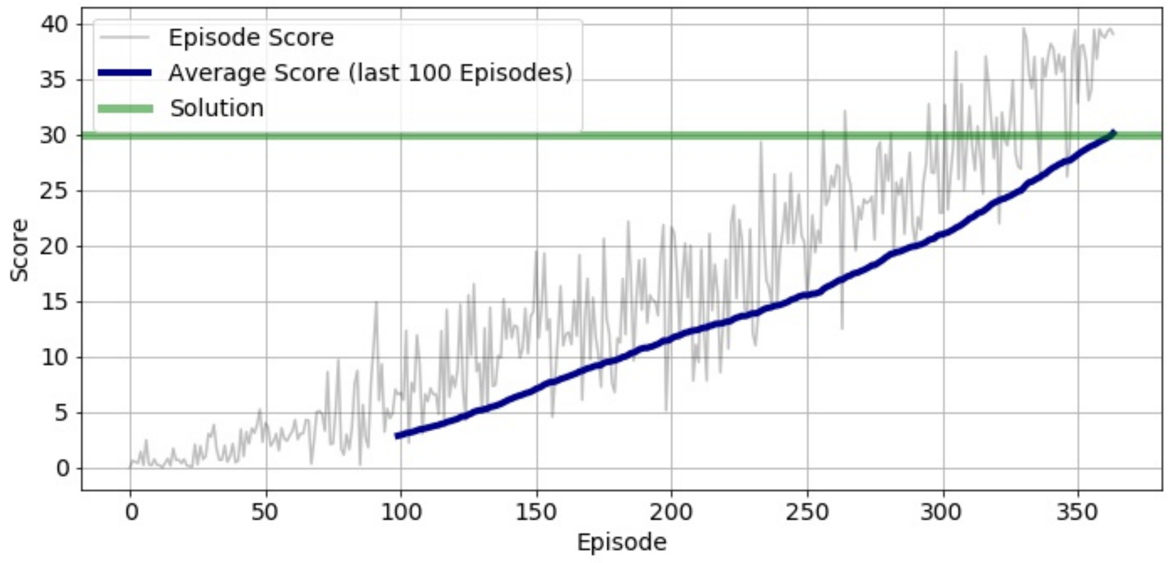
- Experience Replay is also implemented. In this technique, DDPG model is trained by mini-batch from a replay buffer with a size 1e6

- DDPG does not "explore" naturally well, thus a noise function is included. The sigma used is 0.05 based on empirical tests.

- Agent selects next action based on the policy learned by the actor

2. Agent Results:

The following plot of rewards shows the training results. At Episode 363 the algorithm solved the environment.



3. Future Work:

An improvement to be done is to implement multiple agents.

On that solution some tuning on the learning rates and update steps might be done as well.