P2 - Continuous control:

DDPG algorithm is implemented for this project. Learning of continuous space actions requires an actor/model approach. Actor model learns to predict an action vector while the critic model learns Q values for state-action pairs. DDPG uses experience replay (Replay class) to sample batches of uncorrelated experiences to train on.

DDPG agent generates an action for the given state with the online actor model. The model allows the agent to explore different trajentory by adding noise to the action space. The noise is generated through the Ornstein–Uhlenbeck process, which is a stochastic process that is both Gaussian and Markov, drifting towards the mean in long-term.

The agent learns by updating the model thorugh:

- a sample of mini-batches of experiences from the replay buffer to update the value and policy networks.
- all the experiences are saved with (state,actions and rewards,next state) in a finite sized cache, say, replay buffer
- Soft update of the target critic and actor models

Training :

Every timestep a state of the environment is observed.Initially,the agent selects an action for the given state observed from the environment. For the selected action, the environment provides the reward for the action, next state and the information whether the episode is completed.All the experiences are added through the replay buffer with the state,action,reward and next state

Network Architecture:

DDPG uses two network architectures, one for actor and one for critic. The actor network maps state to action and has the following structure: State input (33 units), Network layer is comprised of same number of hidden layers for both actor and critic. The hidden layers with batch normalization speeds up the learning process and also generalises well during training. [1]This batch normalization technique normalizes each dimension across the samples in a minibatch to have unit mean and variance. In addition, it maintains a running average of the mean and variance to use for normalization during testing (in our case, duringexploration or evaluation). Action output (4 units) with tanh activation. The critic network maps state and action to Q with same hidden layers and units with ReLU activation and batch normalization. Final Q-value output is taken from the final output of critic output.

Hyperparameters

With reference to the original paper, the values for the hyperparameters were obtained, various parameters are selected and then implemented based on the results of multiple runs. The most important hyper parameters identified was, standard deviation of the Ornstein–Uhlenbeck process (sigma). It controls the exploration-exploitation trade-off of the action space.

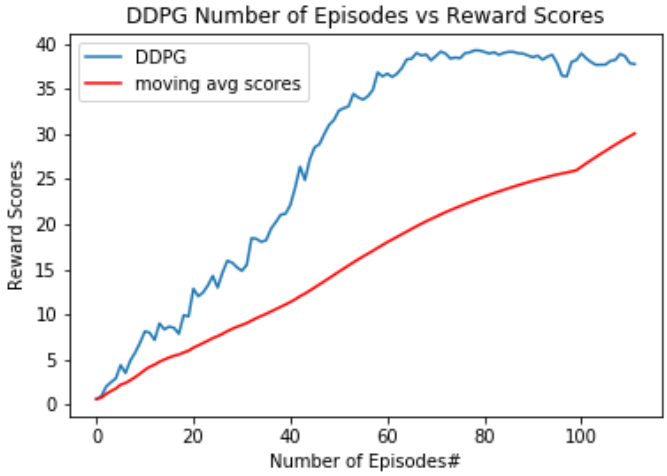
The list of hyper parameters used for this experiment are:

- Replay batch size 128
- Buffer size 1e6

- Replay without prioritization
- TAU from 1e-3
- Learning rate : 1e-3 for both (actor, critic)
- Ornstein-Uhlenbeck noise parameters – 0.2 sigma

Plot of Rewards

The agent was able to solve the environment by achieving score above 30 over 100 consecutive episodes after 117 episodes. There is a slight drop after 60 episodes and the agent stabilies and reaches the expected reward of 30. Plot of rewards can be seen after the environment has been solved. The environment solved in 112 episodes



Ideas for Future Work

- Fine tuning the optimal values for hyperparameters using grid search, beam search or bayesian optimization or genetic algorithm. Like batch normalisation, there are other layer normalisation parameters can be used to experiment in order to speed up the learning process.

- Better control of the exploration-exploitation trade-off by implementing a decreasing schedule for the sigma parameter of the Ornstein-Uhlenbeck process .

- Prioritized experience replay : [2] When using experience replay separates online learning agents and help the agent to process the state transitions in the exact order they are learned. While the prioritized replay further liberates agents from considering the state transitions with random order and experiences are learned by the magnitude of their temporal-differences error.

References:

[1] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2016). Continuous control with deep reinforcement learning. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings.

[2] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 1–21.