

Deep RL Project 3 - Collaboration and Competition

Environment Type: 2 player Tennis environment

Algorithm: Self-play using Deep Deterministic Policy Gradients (DDPG)

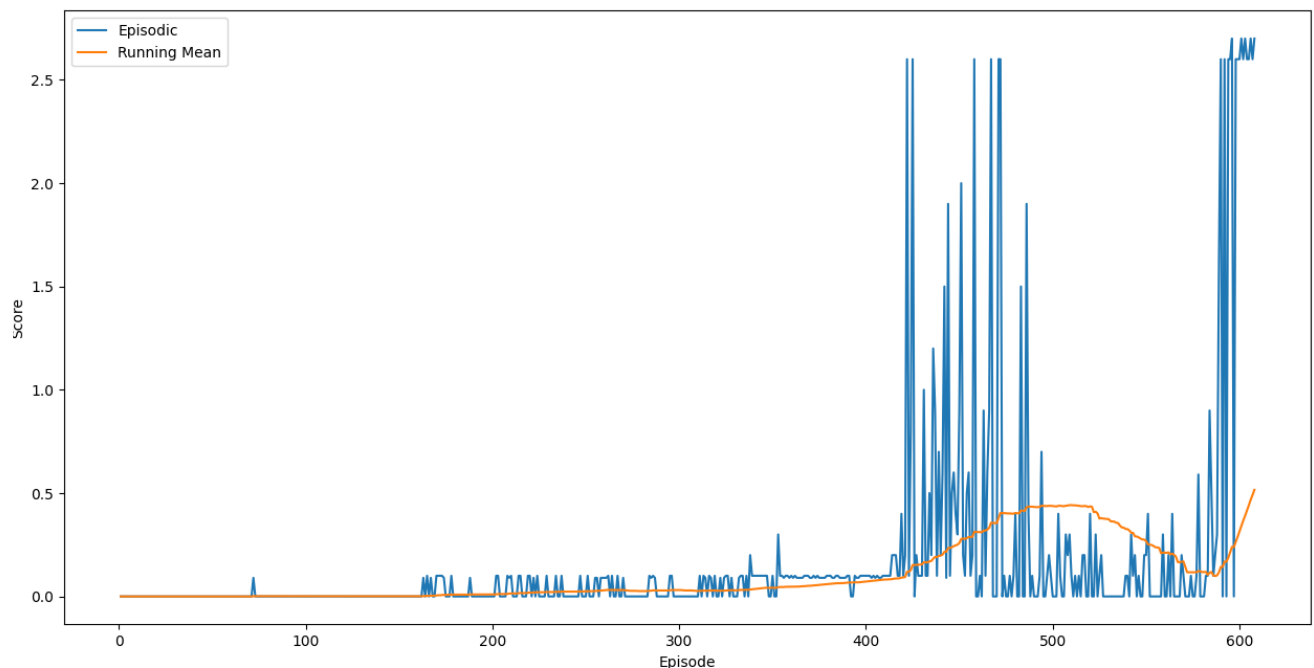
Episodes required to solve environment: 608

Running mean score: 0.516: 30% | 607/2000 [10:27<23:59, 1.03s/it]

Environment solved in 608 episodes! Average Score: 0.52

Note: The running mean score is calculated by averaging the max score between the two agents over the last 100 episodes.

Training Score Plot



DDPG Hyperparameters

Hyperparameter	Description	Value
BUFFER_SIZE	Determines the maximum size of the replay buffer to store past experiences	1e5
BATCH_SIZE	Specifies the mini-batch sized used for learning updates	256
GAMMA	The discount rate on historical rewards	0.99
TAU	The soft update parameter, controlling the degree to which the actor and critic target networks are updated during a learning step	1e-1
LR_ACTOR	The learning rate for the actor model optimiser	2e-4
LR_CRITIC	The learning rate for the critic model optimiser	2e-4
WEIGHT_DECAY	L2 regularisation on model weight parameters to reduce overfitting	0

Neural Network Architectures

Actor

```
self.fc1 = nn.Linear(state_size, fc1_units)
self.fc2 = nn.Linear(fc1_units, fc2_units)
self.fc3 = nn.Linear(fc2_units, action_size)
```

- State_size = 24
- Action_size = 2
- Fc1_units = 256
- Fc2_units = 128
- Activations = ReLU (fc1), ReLU (fc2), Tanh

Critic

```
self.fcs1 = nn.Linear(state_size*n_agents, fcs1_units)
self.fc2 = nn.Linear(fcs1_units+action_size*n_agents, fc2_units)
self.fc3 = nn.Linear(fc2_units, output_dim)
```

- n_agents = 1
- state_size = 24
- fc1_units = 256
- fc2_units = 128 + 2
- Activations = ReLU (fcs1), ReLU (fc2), None

Note: The `n_agents` parameter allows the critic to be repurposed for Multi-Agent DDPG, which is yet to be implemented

Further Work/Improvements

MULTI-AGENT DDPG

The MADDP implementation from Udacity's Competition and Collaboration lab showed the efficacy of decentralised execution and centralised learning. Modifying that implementation for the Tennis environment should be viable, particularly since notebooks/training scripts from other students have demonstrated some success.

PARAMETER SPACE NOISE

My DDPG implementation applies noise to the actions output by the agent's actor network. An alternative would be to use parameter space noise, which has been demonstrated to accelerate learning in experiments conducted by OpenAI. The intuition seems to be that adding noise to the parameters rather than the network output encourages exploration without distorting the relationship between performing an action in a given state and the subsequent observations and rewards.

Source: <https://openai.com/blog/better-exploration-with-parameter-noise/>

EXPERIMENTING WITH NOISE DISTRIBUTIONS

The current implementation uses a Ornstein-Uhlenbeck process to generate noise in the action space. Alternatives such as the Gaussian and Beta distributions may be viable alternatives, with the magnitude of the noise input being decayed over time to shift the algorithm's focus from exploration to exploitation.

DISTRIBUTIONAL VALUE ESTIMATION

The Critic could be redesigned to output a value distribution as opposed to a scalar value. The predicted Q-value would be determined by taking a random sample from said distribution. This would allow the model to be more expressive in its predictions, which may prove useful when encountering state-action pairs with similar (but not identical) representations. A good example of this being implemented for the Tennis environment can be found in the repo linked below.

Source:

<https://github.com/Remtasya/Distributional-Multi-Agent-Actor-Critic-Reinforcement-Learning-MA-DDPG-Tennis-Environment>