Continuous control for robot arm

- Abstract

A DDPG algorithm was used to solve the second project “Continuous control – robot arm” game which was simulated in Unity. The agent contains two parts: actor network and critic network and can attain +30 scores on average after hundreds of episodes. The actor network was built up with simple multi-layers perceptron with two hidden layers, 128, 128 neurons respectively. And the critic network was built up with two hidden layers, 128, 128 neurons respectively. In this project, we also compared if add noise to the action will improve the performance. Our result showed that adding noise generally make the performance more stable than the other. We also found that the DDPG for this game get fluctuated results which may due to 1) shallow networks, 2) compare to states that represented with image and networks that built up with CNN, the vector state and dense networks make the representation change dramatically. The way to improve it might use image-based environment or add batch normalization to hidden layers.

- Project goal

In this project, we’re going to use policy-based or mix way to train an agent due to deep-Q network is unable to solve continuous action space problems. The goal of the agent is to make the joint robot arms reach the object continually.

Method and experiments

Agent information

- State: 33 dimensions include agent's position, rotation, velocity, and angular velocities of the arm.
- Action: 4 continuous actions: control the velocity and angular of two arms.
- Reward: Touch nothing: 0, touch object: +0.1

Model

Main structure: multi-layer perceptron network.

- Actor Network
  - Pytorch solution: 128 →128 → 4
  - Tensorflow solution: 256 → 256 → 4
- Critic Network
  - Pytorch solution: 128 → 128 → 1
  - Tensorflow solution: 256 → 256 → 16 → 1
- Optimizer: Adam, learning rate: 1e-3
- Update per action. Soft-update (switch local net parameters to target net parameters) per 20 steps. TAU is set as 1e-3 (for Pytorch solution and 1e-2 for Tensorflow solution)

Result

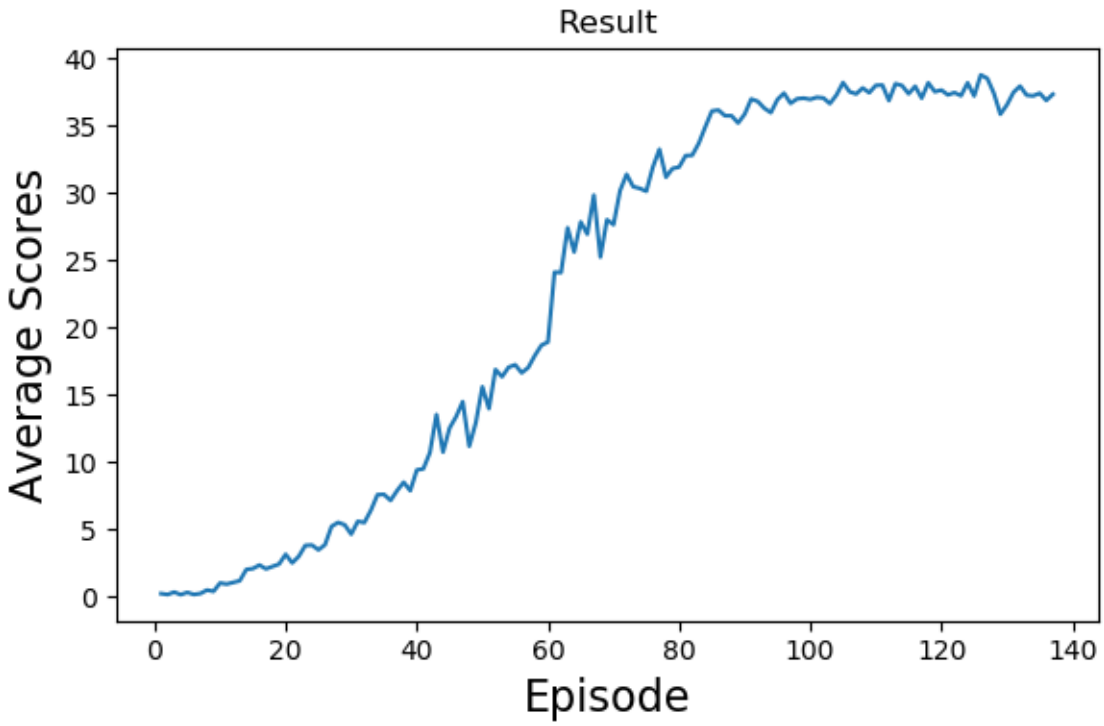

Figure1. Performance with agent that has noise to action.

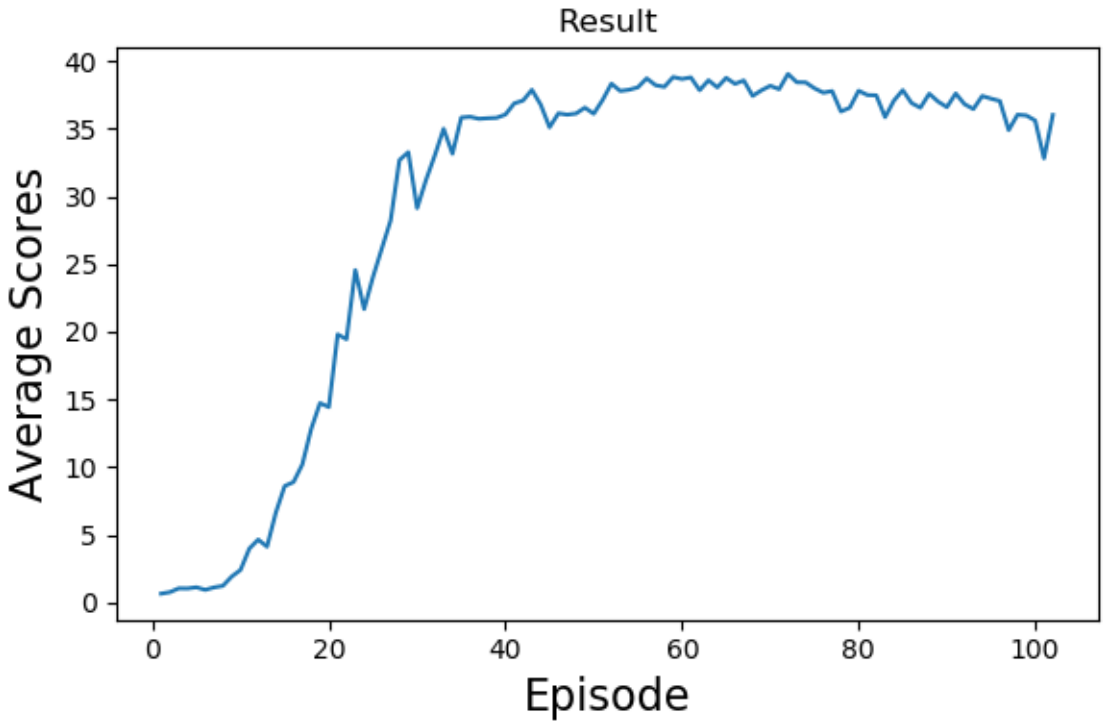

Figure1. Performance with agent that has no additional noise to action.

Discussion

The DDPG is an efficient way to solve the continuous action RL problems. It uses actor network to generate deterministic actions and uses the critic network to evaluate these actions. For the default setting, actions generated from local actor network will become narrow and hard to explore other possibility. With the Ornstein-Uhlenbeck action noise, our results showed that agents' rewards will become more stable. However, we still found that agents' performance seem to highly affected by weights initialization. For some circumstance, rewards will stuck around +15 till the end, while using the same setting but different random seed will get + 30 rewards. It seems weight initialization is a critical issue for deep reinforcement learning while not a critical issue for traditional DNN/CNN for classification or regression tasks. To solve such fluctuation, several method might help 1) use better initialization such as he_norm, 2) use batch normalization or group normalization in hidden layers, and 3) use prioritized sampling to sample batches.