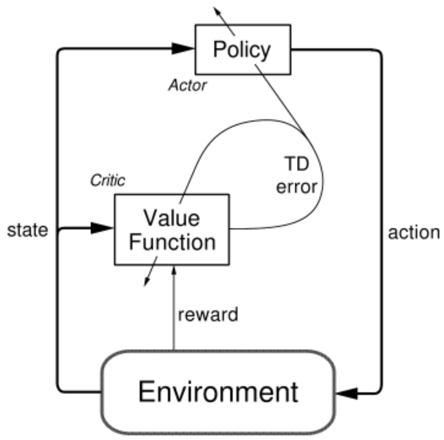
Lab 3(Lab Week 12): Deep Deterministic Policy Gradient

<u>Lab Objective:</u>

In this project, you are going to apply deep reinforcement learning to a continuous action space problem. The training method is deep deterministic policy gradient that involves an actor and a critic. Actor is used to select action whereas critic is used to estimate $Q(s,a)$. You should apply this training process to an environment called Pendulum. Giving current state as input and the return would be the action to perform.

<u>Lab Description:</u>

- Learn how to combine policy gradient with neural network
  - Network design/construction of actor and critic
  - Cooperation between actor and critic
- Understand off-policy reinforcement learning algorithm and the benefits
  - Behavior network and target network
- Implement Deep Deterministic Policy Gradient(DDPG) algorithm
  - Exploring the continuous action space by noise process
  - Understand the difference of deep q learning between DDPG.
  - Update the networks by “soft” target updates



Actor-Critic Architecture

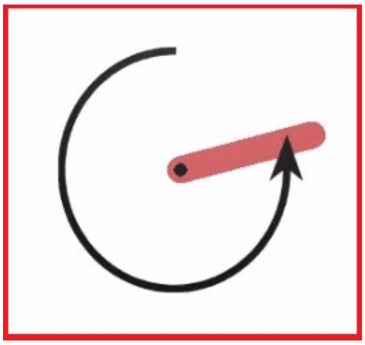
- Apply DDPG to Pendulum environment in OpenAI gym
- <u>Please hand in your source code and report, and demo to TAs.</u>

Environment Setup:

- Tensorflow ≥ 1.0
- Numpy (Would be installed with openai-gym)
- Openai-gym
  - pip install gym[all]
- (optional) xvfb
  - sudo apt-get install xvfb

Game Environment – Pendulum:

- Introduction: The goal is trying to keep a frictionless pendulum standing up.
- State:
  - cos(theta) min: -1.0 max: 1.0
  - sin(theta) min: -1.0 max: 1.0
  - theta dot min: -8.0 max: 8.0
- Actions:
  - Joint effort min: -2.0 max:2.0
- Reward: -(theta^2 + 0.1*theta_dt^2 + 0.001*action^2)

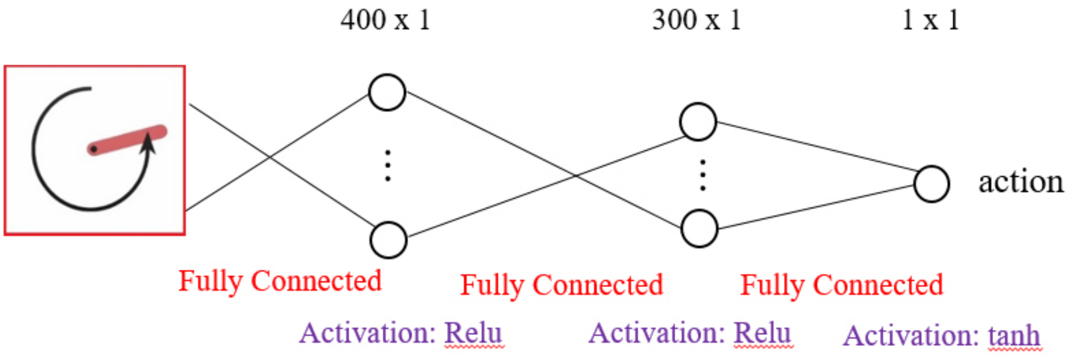


OpenAI: Pendulum

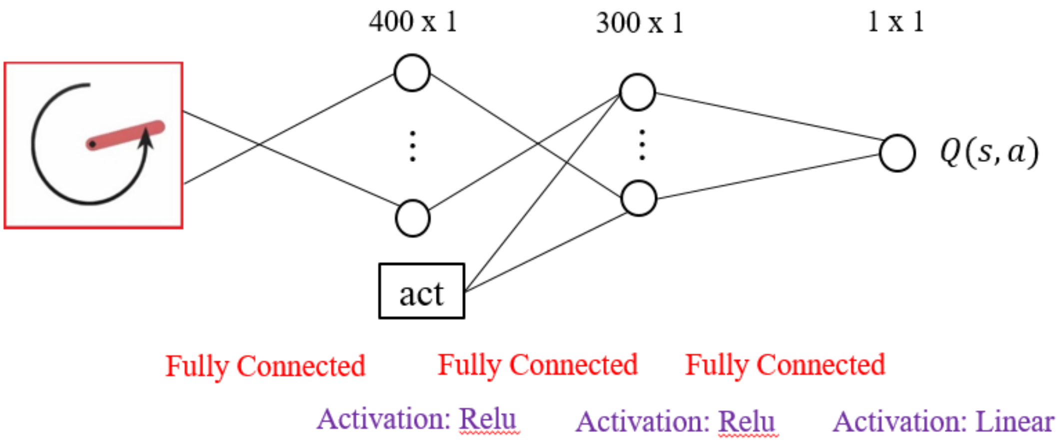
Implementation Details:

Network Architecture

- Actor



- Critic



Training Arguments

- Optimizer: Adam
- Learning Rate
  - Actor: 0.0001
  - Critic: 0.001
- Tau: 0.001
- Batch Size: 64
- Experience buffer size = 10000
- Gamma(Discount Factor): 0.99
- # of training episode: 3500

Misc.

- Training Time: approx. 2~3 hours on GTX960

Requirements:

1. Understand how actor-critic algorithm works.
2. Run deep deterministic policy gradient algorithm in pendulum environment.

Methodology:

Algorithm – DDPG algorithm

Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ

Initialize target network Q' and μ' 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 N for action exploration

  Receive initial observation state s_1

  for $t = 1, T$ **do**

    Select action $a_t = \mu(s_t|\theta^\mu) + N_t$ 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 random minibatch of N transitions (s_j, a_j, r_j, s_{j+1}) from R

    Set $y_i = \ r_i + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right)$

    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:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{\mu'}$$

  end for

end for

Rule of Thumb:

1. The episode score should be able to reach > -1000 after training 40~60 episodes.
2. Don’t set replay buffer size too big. If it costs more memory than your RAM, training process would become very slow.
3. If error “***pyglet.canvas.xlib.NoSuchDisplayException: Cannot connect to None***"” appears, please run the code by following command: ***xvfb-run -s ”-screen 0 1400x900x24” python your_code.py***

Scoring Criteria:

- Report (70%)
  - A plot shows episode rewards of 10000 training episodes (20%)
  - Please explain the mechanism of critic updating(20%)
  - Please describe the algorithm of actor updating(30%)
    - How to calculate the gradients? (Hint: critic, tf.gradients)
    - Explain the corresponding code section that updating actor network.
- Performance – Highest episode reward during training (20%)
  - ≥ -200：100%
  - ≥ -300：80%
  - ≥ -400：60%
  - < -400：0%
- Upload any one video during the training process (5%)
- Upload the last one tensorflow checkpoint file (5%)

References:

[1] (Github) pemami4911/deep-rl

[2] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[3] Lever, Guy. "Deterministic policy gradient algorithms." (2014).