Instruction for q learning

1 Introduction

In this problem, you are asked to implement two classic Reinforcement Learning techniques using deep neural network as function approximation. Deep Q-network is mandatory to implement and Actor-Critic is optional. The environment is 'CartPole-v0' from OpenAI gym. CartPole or 'Inverse Pendulum' as shown in Fig. 1 is a classic optimal control problem. It is such an easy game that the agent only need to balance the pole on top of the little cart as long as possible. You mission is to implement a Reinforcement Learning agent which can keep balancing the pole.

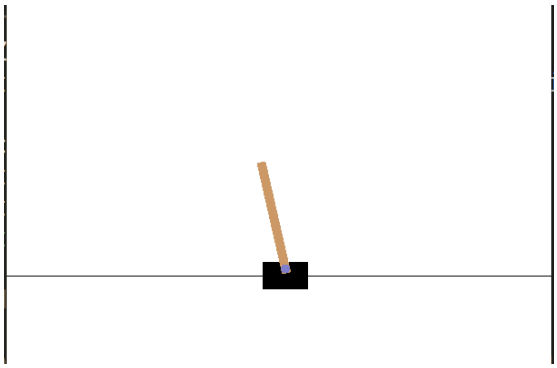

Figure 1: OpenAI gym CartPole-v0

2 Prerequisite

Before starting, you will need to install some softwares. For Ubuntu and macOS users, you only need to follow 2.1. For windows users, please follow 2.2 then 2.1

2.1 Ubuntu & macOS users

First, you need to install OpenAI Gym. Please follow the instruction on https://github.com/openai/gym in order to install Gym. Then please follow

http://pytorch.org/ to install PyTorch.

2.2 Windows users

Since OpenAI Gym and PyTorch are only available on Linux and macOS, you will need to have a virtual machine running Ubuntu first. You may refer to https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html about how to install Ubuntu using vmware14. Then follow 2.1 to install Gym and PyTorch.

3 Implementation

3.1 Deep Q-network (Required)

In this section, you need to implement Q-Learning with neural network as function approximation. The formulation of Q-Learning is shown as Equation 1. Besides Q-learning, you also need to implement Experience Replay and ϵ-greedy to make the whole algorithm work.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{1}$$

where γ is the discounted factor.

3.2 Actor-Critic (Optional)

Actor-Critic methods belong to a broad family of Policy Gradient methods that update the parameters of the policy to optimize the expected cumulative returns in each step. However, Monte Carlo approximates used in vanilla Policy Gradient introduce large sample variance and make the training unstable. To solve these problems, Actor-Critic methods train a value function (critic) jointly with the policy function (actor) and subtract the state value in the policy update step. The actor update rule is as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s_t, a_t)(R_t - V(s_t)] \tag{2}$$

where $V(\cdot)$ is the critic and R_t is the reward to go. The least square error could be used to train the critic.

3.3 Input (Required)

'CartPole-v0' environment in OpenAI Gym only provides a tuple (state, action, reward, next_state, done) as feedback. The observed state is a vector describing current condition of the cart including position, speed, acceleration etc.. It does not provide image as observation. You are required to train two models: the first one taking images as input and the second one taking states as input.

4 Expected Results

We provide the template of DQN in problem2/dqn.py. Please fill the required parts of the code. For Actor-critic, you need to write the code from scratch (optional, no extra credit). You are required to report the reward of each epoch and average reward in a 30 epochs sliding window. The upper limit of duration has been set to 250 steps. You need to report the plot of 200 training epochs as shown in Fig.2. In addition, please write down some empirical comparison between state vector input and image pixel input. Fig.2 is an example, it is not necessary to match that performance. You are welcomed to design your own network architecture and tune the parameter by yourself.

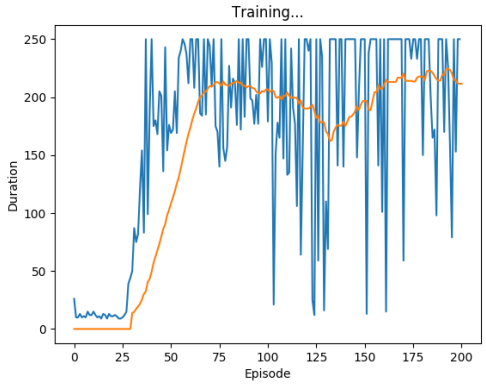

Figure 2: An example of 200 training epochs

What to submit: zip your code and report in one zip file named 'problem2'.