Reinforcement Learning Assignment 4

Yuankun Jiang(TA), Nuowen Kan(TA) Prof. Zou

2019-4-26

1 Introduction

The goal of this assignment is to do experiment with deep Q network (DQN), which combines the advantage of Q-learning and the neural network. In classical Q-learning methods, the action value function Q is intractable with the increase of state space and action space. DQN introduces the success of deep learning and has achieved a super-human level of play in atari games. Your goal is to implement the DQN algorithm and play with it in some classical RL control scenarios.

2 Deep Q-learning

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Figure 1: Deep Q-learning with experience replay

You can refer to the original paper for the details of the DQN. *"Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.*

3 Experiment Description

- Programming language: python3
- You should implement DQN and test it in a classical RL control environment– MountainCar. OPENAI gym provides this environment, which is implemented with python (https://gym.openai.com/envs/MountainCar-v0/). What's more, gym also provides other more complex environment like atari games and mujoco. Since the state is abstracted into car's position,

convolutional layer is not necessary in our experiment. You can get started with OPENAI gym refer to this link (https://gym.openai.com/docs/). Note that it is suggested to implement your neural network on the Tensorflow or Pytorch.

4 Report and Submission

- Every two students form a group and submit a copy of group report and source code.
- Your group report and source code should be compressed and named after "studentID+name+assignment4".
- The files should be emailed to TA (yuankunjiang@126.com) before 2019-05-09 24:00