ELEN E6885: Introduction to Reinforcement Learning
Homework #5

Chenye Yang `cy2540@columbia.edu`

December 3, 2019

P1

Explain why we use "experience replay" and "fixed Q-targets" in DQN. In particular, explain why using "experience replay" and "fixed Q-targets" can help stabilize DQN algorithm when the correlations present in the sequence of observations (e.g., Atari games)?

Ans:
We use experience replay because it can break correlations in data, and brings us back to i.i.d. scenario. It can learn from all past policies.
We use fixed Q-target because it can avoid oscillations and break correlations between Q-network and target.
In Atari games, the DQN training process uses adjacent 4 frames of game screens as the input of the network, and through multiple convolutional layers and fully connected layers, it outputs the Q value of the optional action in the current state to achieve end-to-end learning control.

P2

Recall that DQN is able to stabilize the process of using deep learning for value function approximation, which was believed to be unstable. When it comes to the continuous action space, it is not straightforward to apply DQN directly. One naive approach is to discretize the continuous action space. However, this may not help in practice, why? How does DDPG handle this problem?

Ans:
Discretize the continuous action space may not explore the full state and action space.
DDPG represent deterministic policy by deep network $a = \pi(s, u)$ with weights u.

P3

What is the benefit of using multiple agents in an asynchronous manner in A3C?

Ans:
A3C uses a global network and multiple agents where each has its own set of network parameters.

Each agent interacts with its own local environment and update the global parameter in an asynchronous manner.
Because the samples gathered by an agent are highly correlated and will lead to unstable algorithms if nonlinear function approximation (deep NN) is applied. However, A3C runs several agents in paralleled, each with its own copy of the environment, and use their samples for updates. Different agents will likely experience different states and transitions, thus avoiding the correlation.

P4

Assume Gaussian policy is used in the policy gradient reinforcement learning. The Gaussian mean is a linear combination of state features

$$\mu(s) = \phi(s)^T w = \sum_i \phi_i(s) w_i$$

where $\phi(\cdot)$ represents the vector of feature functions and w represents the weight vector. Also, assume a fixed variance σ^2. Show that that the score function $\nabla_w \log \pi_w(a|s,w)$ of Gaussian policy is given by

$$\nabla_w \log \pi_w(a|s,w) = \frac{(a-\mu(s))\phi(s)}{\sigma^2}$$

Ans:
For Gaussian policy,

$$\pi_w(a|s,w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(a-\phi(s)^T w\right)^2}{2\sigma^2}} \tag{1}$$

$$\log \pi_w(a|s,w) = \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{\left(a-\phi(s)^T w\right)^2}{2\sigma^2} \tag{2}$$

Thus

$$\begin{aligned}\nabla_w \log \pi_w(a|s,w) &= \frac{2\left(a-\phi(s)^T w\right)\phi(s)}{2\sigma^2} \\ &= \frac{(a-\mu(s))\phi(s)}{\sigma^2}\end{aligned} \tag{3}$$

P5

In this problem, you are asked to review AlphaGo paper "Mastering the game of Go with deep neural networks and tree search" and AlphaGo Zero paper "Mastering the game of Go without human knowledge". After the paper review, you need to
(1) Summarize the RL algorithms used in these papers. Especially, point out the differences between AlphaGo and AlphaGo Zero search algorithm.
(2) AlphaGo Zero is self-trained without the human domain knowledge and no pre-training with human games. Explain in detail how AlphaGo Zero achieves self-training.

Ans:
(1)

AlphaGo and AlphaGo Zero use the frame of Generalized Policy Iteration in reinforcement learning. They use Monte Carlo tree search to do both policy improvement and policy evaluation.

In AlphaGo, each edge of Monte Carlo tree search accumulates the visit count and mean evaluation of all simulations passing through that edge

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V\left(s_L^i\right)$$

In AlphaGo Zero, each edge of Monte Carlo tree search stores a set of statistics

$$\{N(s,a), W(s,a), Q(s,a), P(s,a)\}$$

where $N(s,a)$ is the visit count, $W(s,a)$ is the total action-value, $Q(s,a)$ is the mean action-value, and $P(s,a)$ is the prior probability of selecting that edge.
(2)
1. Use MCTS to do policy improvement
At each state s, the policy of the original neural network is $p_\theta(s)$, and the policy obtained through MCTS is $\pi(s)$, so at state s, the policy is improved as: $p_\theta(s) \rightarrow \pi(s)$
2. Use MCTS to do policy evaluation
At each state s, the policy given by MCTS will be executed until the end of the final game, and will be rewarded (according to the game's winning or losing) z, as the evaluation of policy.
3. Train Neural Network We can collect training data (s, π, z) for each experiment. According to the training data, the following loss function can be constructed:

$$l = (z - v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c\|\theta\|^2$$

which is used to update the weight of NN through supervised learning after each iteration.