ELEN6885 Reinforcement Learning HW5

Tong Wu, tw2906

Problem 1

Because the experience replay can step out of the correlation that comes with data, which is called independent and identically distributed random variables environment. The fixed Q-targets then, can also step out the correlation between action values and the target, and stabilise the algorithm of DQN.

In sequence of observations like Atari games, the DQN is prefer to forgot the previous situation, so the experience replay can be useful to avoid this case and maintain the previous situation. Fixed Q-targets can replace the attributes which are not good and replace by the new network, which can make the training easier.

Problem 2

Because the discretion of the continuous action space needs to maximise the Q value for each step, which is performance-costing and give processor too much pressure. Also, the maximising the Q value for each step needs a lot of storage space, to make the computing more complicated, which makes the continuous space may not be able to explore all states.

DDPG is the fusion of the DPG and DQN, which presenting as a off-policy algorithm, achieved by deep network, so it can handle with this problem.

Problem 3

The multiple agents in A3C uses local environment to communicate with each other, and are able to update the global variable in an asynchronous manner in A3C. This feature brings the stabilise to the training process. Also, since A3C runs multiple agents in parallel, and each agent has its own local environment which is isolated with others, and each agents can keep updating via samples. So these makes each agents experiences at different environment with different states and rewards, in order to avoid the correlation with each other.

Problem 4

According to the Gaussian policy, where

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

So the log transform should be:

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

Then:

$$\begin{aligned}\nabla_w log\pi_w(a|s,w) &= \nabla_w(log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(\alpha-\phi(s)^T w)^2}{2\sigma^2}) \\ &= \frac{2(\alpha-\phi(s)^T w)\phi(s)}{2\sigma^2} \\ &= \frac{(\alpha-\mu(s))\phi(s)}{\sigma^2}\end{aligned}$$

Problem 5

Part 1

Alpha Go's architecture has two layers, which are policy network and value network. In policy network, the purpose is to decide the next best move, with 13 hidden layers of CNN [1][3]. The policy network has two stages, each are supervised learning to predict human expert moves, and improve network by policy gradient reinforcement learning [1][3]. The value network has purpose to evaluate the chances of winning, using 14 hidden layers of CNN, train the network by regression on state-outcome pair sampled from self-play data using policy network [1][3]. Once the model has been trained, using MCTS (Monte-Carlo Tree Search) to combine the policy networks and value networks to select actions by lookahead search [1][3].

AlphaGo Zero trained itself by self-play only without any supervision. It has a single neural network with ResNets structure, which can decide the next best move and evaluate the chances of winning [2][3]. Then, AlphaGo Zero uses simpler tree search to complete the evaluation of positions and moves [2][3].

Part 2

The neural network in AlphaGo Zero is trained from games of self-play by a novel reinforcement learning algorithm. At first, the neural network is initialised to random weights [2]. At each subsequent iteration, games of self-play are generated [2]. Then, for each time-step, MCTS search will be executed by using the last iteration result of neural network and a move is played by sampling the search probabilities [2]. When the search value drops below a resignation threshold or when the game exceeds a maximum length, the game will terminated, and the game will then scored to give a reward [2]. The data for each time step is stored. In parallel, new network parameters are trained from data sampled uniformly among all time-steps of the lasty iterations of self-play [2]. The neural network is adjusted to minimise the error between the predicted value and the self-play winner, and to maximise the similarity of the neural network move probabilities to the search probabilities [2].

References

[1] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* , *529* (7587), Article 7587. https://doi.org/10.1038/nature16961

[2] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature* , *550* (7676), Article 7676. https://doi.org/10.1038/nature24270

[3] Chen, H. (n.d.). University of Waterloo, CS885 Reinforcement Learning. *Mastering the Game of Go without Human Knowledge* .