Navigation using Deep Q-Network

Laha Ale
Department of Computing Sciences
Texas A&M University Corpus Christi
Corpus Christi, USA
Email: lale@islander.tamucc.edu

***Abstract*—Deep reinforcement learning, as the most advanced artificial intelligence method, mimic human intelligence and behaviors to interaction with the world. In this work, we try to train a model, using deep reinforcement learning, to control game agent of The Unity Machine Learning Agents Toolkit (ML-Agents)[1]. Precisely, the agent gets a reward if it collects a yellow banana or a punish if it obtains a blue one. We adopted the deep Q-network method to learn and control the agent. The results show the agent can achieve human-level performance. The code of this project has housed on GitHub[2].**

***Index Terms*—Navigation, Deep Q-Network, Deep learning,Deep Reinforcement Learning**

I. Learning Algorithm

As shown in Alg.1, there two significant steps in the algorithm, it takes action and stores the learned information in the replay buffer; further, sample and learns from the stored data, also known as experience reply.

The hyperparameters are set as follow. There are three layers in the Deep Q-Network [1]. In the first layer, a fully connected layer with input size 37 (state size) and output 64. In the second layer, another fully connected layer input size 64 and output 64. In the last layer, it is a fully connected layer with input 64 output 4 (action size).

Besides, the hyperparameter of the DQN setting as follows: the values of maximum steps per episode, start ϵ, end ϵ, and ϵ decay rate set as 1000,1.0,0.01, and 0.999. Another parameter for base score is parameter is 13.0; the DQN algorithm will quite once the it pass this score.

Algorithm 1: Deep Q-Learning

Initialize replay memory D with capacity N;
Initialize action-value function $\hat{q}$ with random weights W;
Initialize target action-value weights $W^- \leftarrow W$;
for *the episode* $t \leftarrow 1$ *to* T **do**
  Initial input frame x_1;
  Prepare initial state:$s \leftarrow \phi(< x_1 >)$;
  for *time step* $t \leftarrow 1$ *to* T **do**
    //Sample
    Choose action A from sate S suing policy $\pi \leftarrow \epsilon - greedy(\hat{q}(S, A, W))$;
    Take action A, observe reward R , and next input frame x_{t+1};
    Prepare next state: $s' \leftarrow \phi(< x_{t-2}, x_{t-1}, x_t, x_{t+1} >)$;
    Store experience tuple (S, A, R, S') in replay memory D;
    $s \leftarrow s'$;
    //Learn
    Obtain random mini-batch of (s_j, a_j, r_j, s_{j+1}) from D;
    Set target $y_i = r_i + \gamma max_a \hat{q}(s_{j+1}, a, W^-)$;
    Update:$\Delta W = \alpha(y_i - \hat{q}(s_j, a_j, W)) \bigtriangledown_W \hat{q}(s_j, a_j, W)$;
    Every C steps, reset:$W^- \leftarrow W$;

II. Plot of Rewards

As we can see from Fig.1, the average score roughly is equal or above 13.0 after 300 episode training.

III. Ideas for Future Work

The neural network in this work is a rather shallow and standard version; we can adopt more sophisticated architecture in the future work that probably improves the performance.

Besides, DQN also can be improved by adopting the following possible methods. First, Double DQN introduce in [2] to address deep Q-learning tends to overestimate action values [3]. Second, prioritized experienced replay presented in [], can improve the experience replay process; further, it can enhance the performance of the model. Last, Dueling DQN introduced in [4], is another powerful method that allows us to assess the value of each state, without having to learn the effect of each action.

[1]https://github.com/Unity-Technologies/ml-agents
[2]https://github.com/ainilaha/Navigation-Banana

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[2] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[3] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School*, D. T. J. E. M. Mozer, P. Smolensky and A. Weigend, Eds. Erlbaum Associates, January 1993.

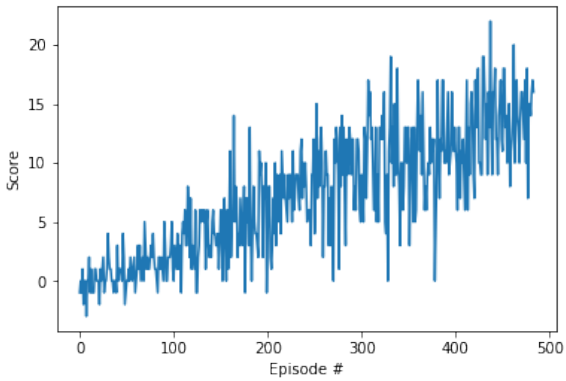

Fig. 1. Results

[4] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015. [Online]. Available: http://arxiv.org/abs/1511.06581

[5] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2015. [Online]. Available: http://arxiv.org/abs/1511.05952