Report for Project 1 – Navigation

Siqi Bai

Problem

In this project, the goal is to train an agent to navigate through a 2D space, while collecting as much yellow bananas as possible and avoiding blue bananas in the same time. The state is 37 dimensional and the actions have 4 options and are discrete. The criteria for solving the problem is to achieve +13 scores averaged over the latest 100 episodes. The maximum allowed episode number is 1800.

Approach

To solve the problem, I use a value-based approach to learn the optimal action value function by training a deep neural network, which has been successful in training an agent to master challenging video games, such as Atari [1]. The algorithm uses experience replay to achieve high data efficiency and break the correlation. Also, when calculating the weight update, the network uses a separate network to compute the target in the Q-learning update to make the learning process more stable. In this project, we largely adopt the above approach.

The deep neural network has a multi-layer feedforward structure, consisting of 7 hidden linear layers and one output layer. The numbers of hidden neurons in each layer are 64, 128, 128, 128, 64, 64, 32. The final output layer has 4 neurons. The activation of each hidden layer is followed by Relu nonlinearity.

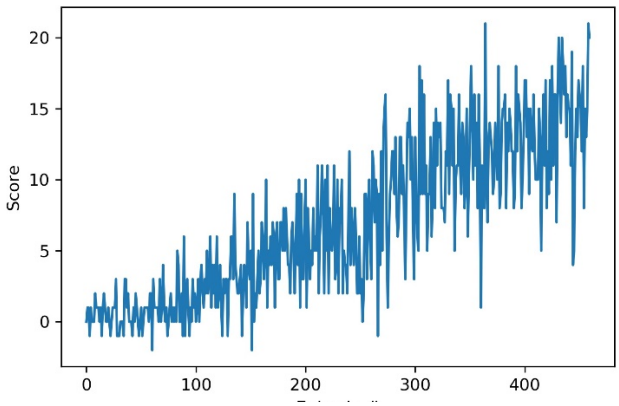
To train the deep neural network, I implement a deep Q-learning algorithm that iteratively updates the network weights using the experience collected by the agent. The weight update rule is shown in Equation (1). The hyper parameters are chosen as the following: the learning rate α=5e-4, the reward decay factor γ=0.99. The weight w^- is soft-updated every 4 timesteps following the equation (2), where τ=1e-3. To break the correlation between adjacent experiences, we use a memory buffer that has a maximum of 60000 timesteps. For each weight update, a minibatch of 64 experiences $(S_t, A_t, R_{t+1}, S_{t+1})$ are sampled from the memory buffer.

$$\Delta w = \alpha \left(R + \gamma \max_a q_{target}(S', a, w^-) - q_{local}(S, A, w) \right) \nabla_w q_{local}(S, A, w) \quad (1)$$

$$w^- = \tau w + (1 - \tau) w^- \quad (2)$$

Result

By implementing the algorithm described above, I was able to solve the problem in ~360 episodes. The scores are plotted as a function of the episodes, as shown in the figure below.



Future directions

In the future, the following approaches can be tried to further improve the performance of the learning algorithm and the agent.

First, the hyperparameters can be further optimized, such as the hidden layers and the number of hidden neurons in the deep neural nets, the size of memory buffer, the minibatch size, the epsilon decay rate, etc. For the neural networks, if the structure is too simple, it may not be able to learn the optimal action value function. On the other hand, if the network is too big, it may lead to overfitting and performs badly in generalization. The memory buffer size will control how far the previous experiences can affect the current learning. Regarding the minibatch size, a too small value will result in stochastic weight update that maybe difficult to converge. Too large minibatch size will result in low learning efficiency and high computational expense. For the epsilon decay rate, it directly affects the actions that the agent takes at each time step. Too large epsilon decay rate will trap the agent in local maximum, resulting in suboptimal policy. Too small epsilon decay rate will make the agent behavior random for long time steps and prevent learning.

Second, better algorithms can be implemented, such as double DQN [2], prioritized experience replay [3], and dueling DQN [4]. The double DQN has been shown to alleviate the overestimation problems in the DQN algorithm. The prioritized experience replay uses the TD error as weights to change the sampling probability of the experiences to emphasize important ones. The dueling DQN tries to estimate two separate estimators for the state value function and the state-dependent action advantage function. Then it determines the action value function by adding them up. All the above three improvements can be tried here to further improve the agent's performance.

Reference

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), p.529.

[2] Van Hasselt, H., Guez, A. and Silver, D., 2016, February. Deep Reinforcement Learning with Double Q-Learning. In AAAI (Vol. 2, p. 5).

[3] Schaul, T., Quan, J., Antonoglou, I. and Silver, D., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952.

[4] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M. and De Freitas, N., 2015. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581.