AE 598 RL HW 2: DQN

Grace E. Calkins*

I. Code Structure

This writeup presents the results of Homework 2 for Reinforcement Learning (AE 598 RL), where I wrote a DQN algorithm.

My code was structured with the DQN method and associated objects in `dqn.py`, which was imported to `train_discreteaction_pendulum.py` to train. Helper functions, like a function to select an action following ϵ-greedy and plotting functions, are also in separate files. The `main` function in the training files has a series of boolean tags at the beginning which can be changed to run different scenarios.

The DQN algorithm consists of a neural network `DQN` object, an episode replay `ReplayBuffer` object, and three functions:

- `fill_replay_buffer`, which fills the initial replay buffer with random actions,
- `update_Q_weights`, which performs optimization on the weights of the DQN neural network
- `DQN`, which runs the DQN algorithm once, and
- `DQN_avg`, which runs the DQN algorithm with the same hyperparameters multiple times to outputs a list of results for post-processing statistics.

In `train_discreteaction_pendulum.py`, the environment, log of episode returns, and the trained DQN agents are saved to the `data` directory via `pickle`. The data can then be easily loaded into plotting functions to compare with other trained agents. There are a series of boolean parameters in the preamble to turn on and off certain sections of the analysis.

II. Results

The results presented below use the average of 5 DQN trained agents to compute the optimal action or state value function. The state is passed into the helper functions `avgDQNsa` and `avgDQNsV`, which first compute the mean Q vector by averaging the results from each trained agent, and then compute the argmax or max of the mean Q vector. This was done to show trends more clearly, as each trained agent is unique. The hyperparameters chosen are presented in Table 1. They are largely based off the values from Extend Data Table 1 of [1]. Some parameters, like replay size and the episode number and reset rate, were decreased as our model runs for fewer episodes. 100 episodes were used as running more episodes would be computationally expensive, and the agent was able to balance the pendulum robustly (reach vertical quickly from a variety of initial conditions) in 100 episodes. ϵ was decayed linearly from the initial value to the final value over all episodes in the training. The replay buffer had an initial size of 500 random transitions, and as new transitions were added that caused the replay buffer to exceed the specified size, the oldest transitions were removed. A neural network with 64 units per layer was used, with `tanh` activation on the two hidden layers and a linear activation function on the output layer was used for the target and policy Q networks.

Figure 1 shows the learning curve (average episode return) for 5 trained agents over 100 episodes. Each dot represents a data point from the 5 trained agents, and the red lines show the mean and 1σ standard deviation from the 5 runs. As expected, the mean grows throughout time as the agent is learning the optimal policy.

*Master's Student, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, 104 S Wright St, Urbana, IL 61801

Table 1 Hyperparameters for DQN.

Hyperparameter	Value
Replay Size	100,000
Learning Rate (α)	0.00025
Discount Factor (γ)	0.95
Initial Epsilon	1
Final Epsilon	0.1
Num. Episodes	100
Weight Update Batch Size	32
Num. Episodes Between Target Network Updates	10

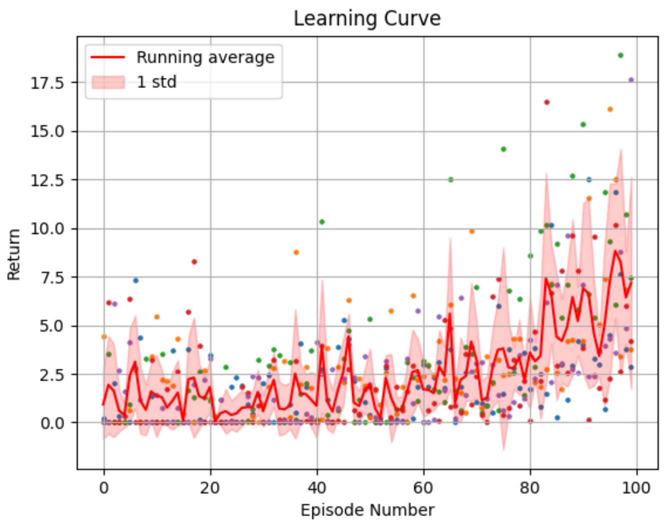

Fig. 1 Average learning curve for 5 trained agents

Figure 2 shows trajectory for the trained agents. As expected, the agent is able to balance the pendulum vertically very soon after the episode starts. A gif of the trained agent can be found in my rep at `https://github.com/uiuc-ae598-rl-2023-spring/hw2-dqn-calkins7/blob/main/my_figures/hw_final__trained_pendulum.gif`.

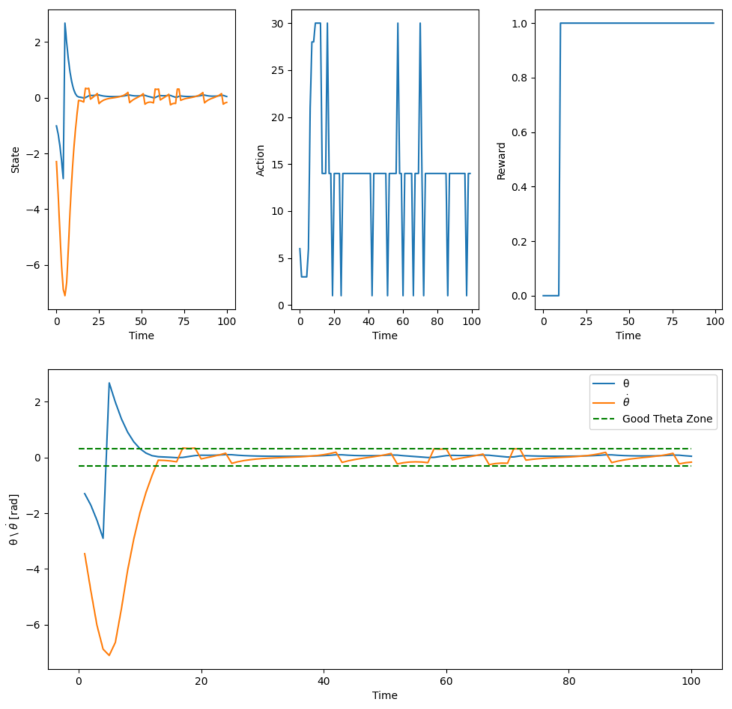

Fig. 2 Example trajectory for 5 trained agents

Figure 3 shows the optimal policy for the trained agents. 0 corresponds to a torque of -4, while 32 corresponds to a torque of 4.

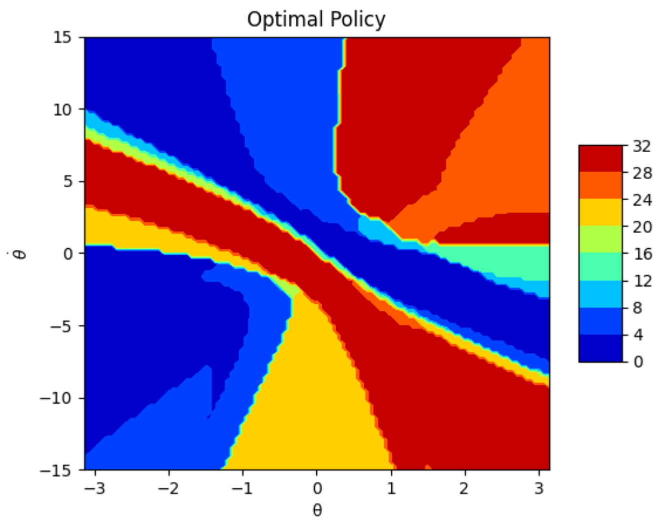

Fig. 3 Optimal policy for 5 trained agents

Figure 4 shows the optimal state value function for the trained agents. The highest state value occurs near $\theta = 0$, as seen by the red on the plot. The negative slope of the value function contour relative to $\dot{\theta}$ and θ also makes sense. There is a greater reward to being left of vertical (negative θ) and having a positive $\dot{\theta}$ than for being left of vertical and having a negative $\dot{\theta}$ because the former is moving towards the desired equilibrium point and the latter is moving away from the equilibrium point.

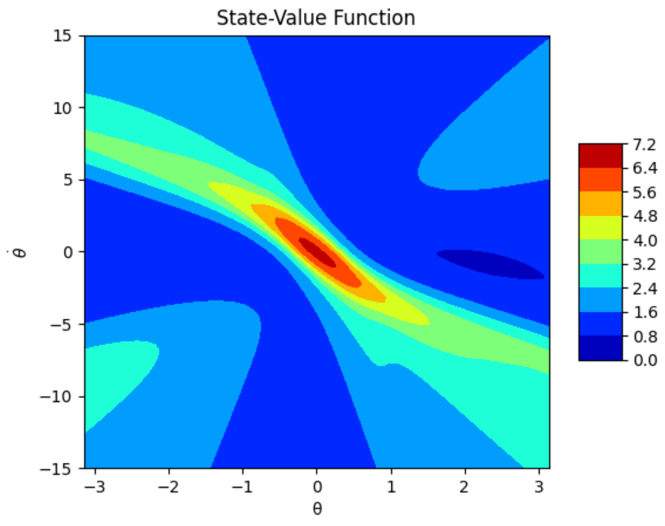

Fig. 4 Optimal state value function for 5 trained agents

The results of the ablation study, which varied whether the target network was on and if the replay buffer was used, are presented in Figure 5. The same hyperparameters in Table 1 are used, but to turn off the replay buffer, the replay size is set to 1. The target Q network is turned on and off from a boolean passed in the DQN method. The results shown are the average of 5 trained agents for each case. DQN outperforms the other methods drastically, with the other methods seeing no trending increase in average return even after 100 episodes.

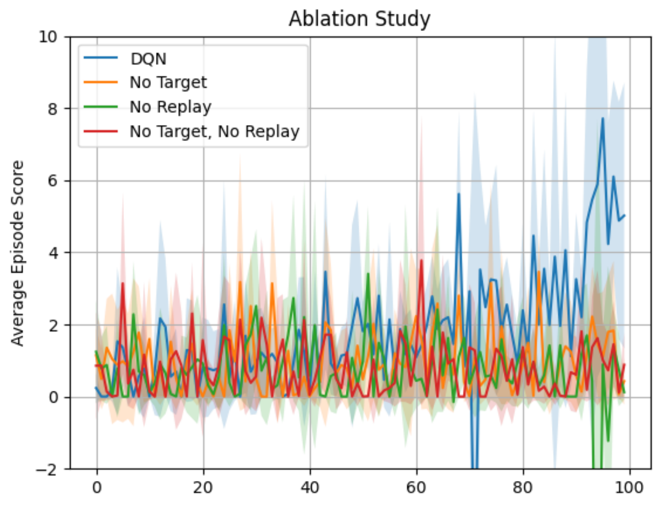

Fig. 5 Ablation study results

III. Conclusions

These results show that DQN is a very successful method for train an agent to balance a pendulum vertically. It is an improvement over the Q-learning implemented in our last HW as it used a neural network to approximate Q, enabling continuous state space. The ablation study results show that using the target network an episode replay are vital for DQN performance.

References

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