AE598 HW2: Deep Q-Network

I. Introduction

Deep-Q network (DQN) is a popular algorithm in reinforcement learning that resembles the model-free Q-learning, with the exception that the agent has to use a deep neural network to approximate its Q-value (state-action value) rather than being rewarded a score based on a finite table. In DQN, the input to the deep neural network is the state or observation vector and the number of neurons in the final output layers is the number of actions the agent can take. Thus, the training target is the Q-value of each action the agent takes, and the input is the state the agent is in.

In mathematical language, we aim to train a policy to maximize the cumulative discounted rewards:

$$R_{t_0} = \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t \tag{1}$$

Thus, our goal is to find a policy such that :

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q^*(s,a) \tag{2}$$

where $Q^*(s,a)$ maps a state-action pair to a scalar value. Since we don't have access to $Q^*(s,a)$, we create a deep neural network to approximate it. We first create a temporal difference error term defined as :

$$\delta = Q(s,a) - \left(r + \gamma \max_{a}^{'} Q\left(s',a\right)\right) \tag{3}$$

then, a loss function for training can be constructed using a Huber loss or mean-squared error loss. Effectively, we are learning the parameters (weight vector) that constitute our neural network to approximate Q

II. Balancing Cartpole

In this assignment, we apply the aforementioned algorithm to control the swing-up motion of a classic cart-pole. We are given a continuous state space and a discretized action space. It is known that the advantage of applying DQN is the ability to handle continuous state spaces.

III. Results

Across all training episodes, the state trajectory is shown in Fig. 1

A. Learning Curve

The payoff as a function of training steps is shown in Fig . 2

B. Trained Policy

A heatmap for the trained policy shows the relationship between the state space and the action taken by the trained agent. See Fig. 3

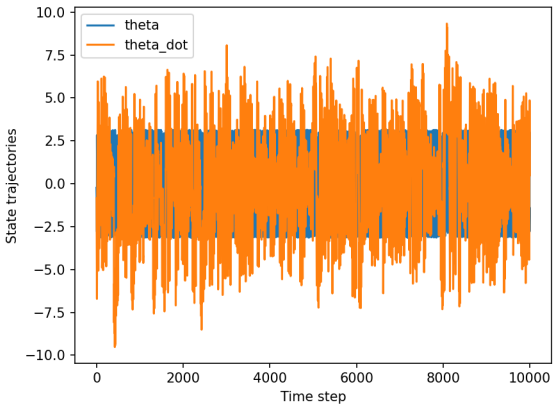

Fig. 1: Full State Trajectory

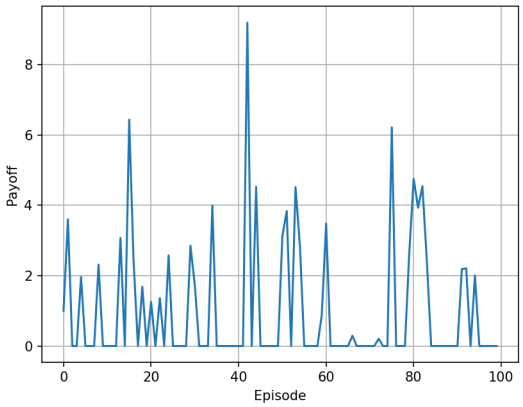

Fig. 2: Learning Curve

C. State Values Prediction

A heatmap for the predicted states values is shown in Fig.4

D. Ablation Study

An ablation study is performed for three additional scenarios: DQN without a target, DQN without replay, and DQN without a target and replay. The learning curves for all scenarios are shown in Fig .5:

IV. Discussion

For all the training scenarios, a batch size of 64 is used, and the deep neural network used in training consists of 2 hidden layers, each activated by a $tanh$ function with 64 neurons. A discount factor γ = 0.95 is used. The weights of the target

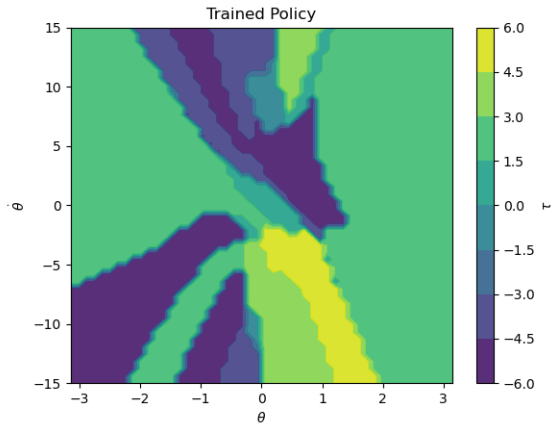

Fig. 3: Trained Policy

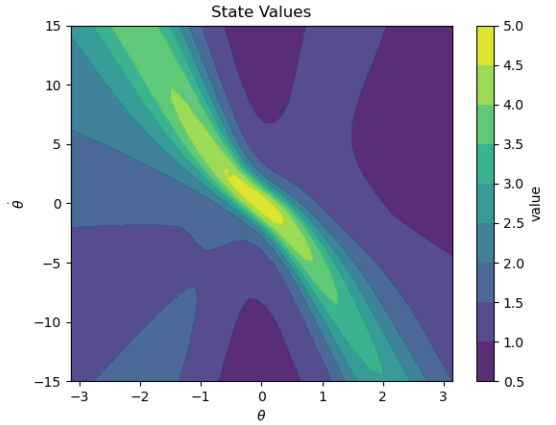

Fig. 4: State Values

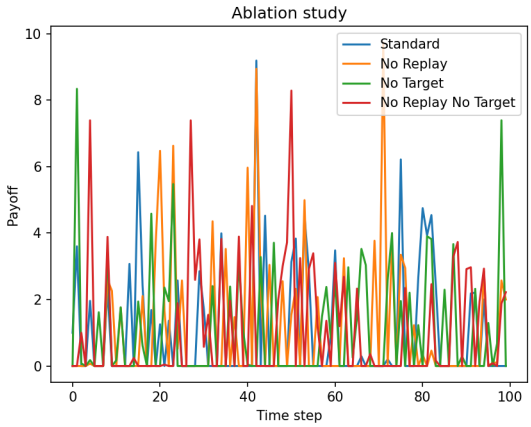

Fig. 5: Learning Curve for Ablation Study

network and the policy (online) network are synchronized once every 1000 training steps. Additional hyperparameters include the learning rate used in the backpropagation solver in Pytorch – the RMSprop solver is used.

However, according to the learning curves shown above, the swing-up motion was not achieved after training 100 episodes. After double-checking the code, no error is found, and therefore the exact reason that has caused the divergence swing-up control is not known yet. A hypothesis is that the decay of ϵ has an impact on the learning outcome, but after changing to a few different ϵ decays, the payoff of DQN still wouldn't show an improving trend.

For the standard DQN algorithm, we see in Fig. 3 that the action torque τ is positive even when θ is positive, which is not reasonable. The learning curves in the ablation study do not showcase a significant differences between each scenario as the the policy learned is not able to control the swing-up motion of the pendulum perfectly.

To summarize the performance of the ablation study, the averaged payoff from each scenario is given in the following table:

	Average Payoff
Standard DQN	1.001
No Replay	1.098
No Target	1.047
No Target, No Replay	1.106

We conclude that the algorithm as it is implemented right now has no significant errors, but the tuning of the right combination of hyperparameters is not found yet after spending a significant amount of time, which has caused an inaccurate action policy to be returned – intuitively, the learning curve should clearly show an trend of improving payoff over training steps, and the action policy should ouput a restoring torque that counters the motion of the pendulum.