Imperial College
London

COMP70028

Reinforcement Learning

October 2023

Imperial College London

Department of Computing

—Coursework 2—

Name:
Kyoya Higashino

Course:

MSc Artificial Intelligence

1 Problem Description

Problem description

Your goal is to train an agent to balance a pole attached (by a frictionless joint) to a moving (frictionless) cart by applying a fixed force to the cart in either the left or right direction. Please see Fig. 1 for an illustration. The aim is to train the DQN to keep the pole balanced (upright) for as many steps as possible. We do not control the magnitude of force we apply to the cart, only the direction. The optimal policy will account for deviations from the upright position and push the cartpole such that it remains balanced.

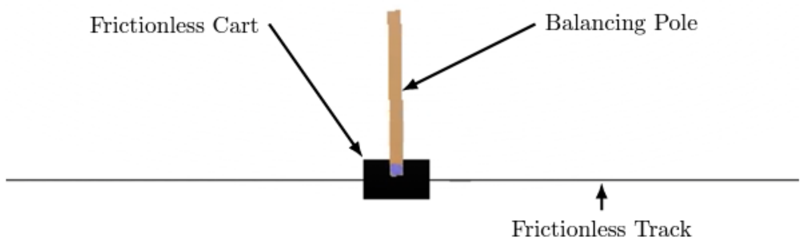


Figure 1: Illustration of the OpenAI Gym CartPole environment.

Our action space is discrete and of size 2. We have action 0, apply a force on the cart to the left, or action 1, apply a force on the cart to the right. The observation state we obtain from the environment is of size 4 of the following positions and velocities: cart position, cart velocity, pole angle and pole angular velocity.

All four observations of the environment are initially assigned a random value between -0.05 and 0.05. So the cart starts close to the origin with the pole almost upright and a low initial angular velocity. The aim is to keep the pole upright for as many steps as possible. This makes the reward quite simple to define: for each step taken (including final step) a reward of $+1$ is returned. The environment will only terminate when it reaches any of the following states:

- pole angle greater than $\pm 12°$ (or equivalently 0.2094 radians)
- cart distance from centre greater than ± 2.4,
- or the number of steps exceeds 500.

2 Tuning the DQN

2.1 Hyperparameters

Using iterative grid searches to run experiments guided by DQN theory, many hyperparameters were explored, both in isolation and in combination with other parameters. Tables 1 and 2 below detail the hyperparameter search space as well as the final hyperparameters chosen via empirical evidence and the sum of all rewards over all 10 runs as the evaluation metric.

Table 1: Hyperparameter Search Space

Hyperparameter	Values Explored via Grid Search
*Number of Hidden Layers	1, 2, 3
*Number of Neurons per Hidden Layer	5, 10, 15, 18, 20
Batch Size	100, 300, 500, 700
Replay Buffer Size	9000, 12000, 15000, 18000
Learning Rate	0.001, 0.005, 0.01, 0.02
Optimiser	SGD, Adam, Adagrad
*Epsilon Decay Range	(0 → 1), (0.1 → 0.9), (0.2 → 0.8)
*Epsilon Decay Duration	150, 200, 250, 300
*Sampling Technique	Default, Adaptative

*Note that the number of hidden layers and neurons are later combined into one hyperparameter called "Architecture" and epsilon decay range and duration are combined into "Epsilon Decay Configuration".

*"Sampling Technique" refers to the sampling method before the replay buffer storage reaches the batch size. "Default" means no sampling is performed while "Adaptive" means the entire buffer is sampled.

Table 2: Final Hyperparameters

Hyperparameter	Value	Justification Summary (see below for more detail)
Architecture	[4, 18, 18, 2]	As the cart-pole problem is relatively simple, simple architectures were theorised and shown to be most effective. The optimal architecture was eventually found through an iterative grid search.
Batch Size	500	As 'successful' episodes generate states from 100-500, the proposed reasoning was that samples should be of comparable size, sampling at least one episode's worth of experiences. The optimal batch size was found eventually through a grid search.
Replay Buffer Capacity	15000	The buffer capacity needs to be large enough to remember valuable past experiences while small enough not to severely delay the learning process. The optimal balance was eventually found through consideration of the batch size and grid search.
Learning Rate	0.005	High learning rates facilitate fast learning, but low rates ensure stable convergence and prevent overshoot. The optimal balance was eventually found through a grid search.
Optimiser	Adam	As gradients change over the course of training, adaptive optimisers like Agagrad and Adam were shown to perform better than normal Stochastic Gradient Descent. The optimal optimiser was found through a grid search.

Epsilon Decay Configuration	[1, 0, 250]*	To promote exploratory behaviour at the start of runs but exploitative actions as experiences increase, epsilon should decay over each episode. Furthermore, the varying start and end epsilon values should change model performance, being able to maximum and minimum level of exploration possible. Through linear decay, the optimal starting, end, and decay duration values were found through a grid search. *Read as [start, stop, decay length]
Sampling Technique	Adaptative	The default code sampled from the replay buffer only when the number of experiences equalled the batch size. This approach delayed learning until a full batch of experience was accumulated. To address this, when the desired number of samples exceeds the experiences in the replay buffer, the entire buffer is sampled (adaptive sampling). This was also practically demonstrated through repeated experiments.

After selecting the final hyperparameters, a random search was conducted within the hyperparameter search space to explore potential alternative models. However, none of the randomly generated models demonstrated superior performance compared to the final hyperparameter combination, further reinforcing that the selected hyperparameters were the optimal choice.

While many hyperparameters were tuned individually, the effects of the batch size and replay buffer capacity are plotted in Figures 1 and 2 below. As both parameters are unique characteristics of DQNs that have not been explored in previous coursework, they are observed in greater detail below.

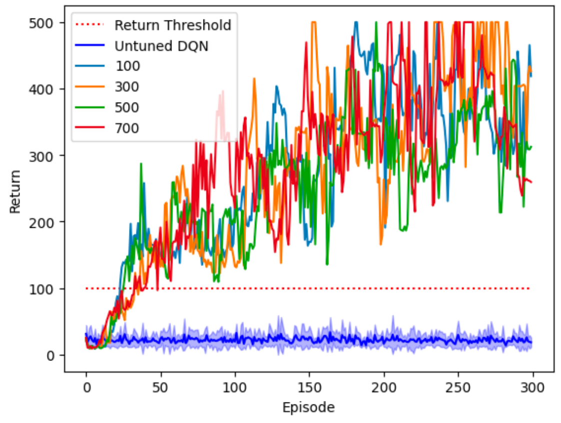


Figure 1: Effect of varying batch size

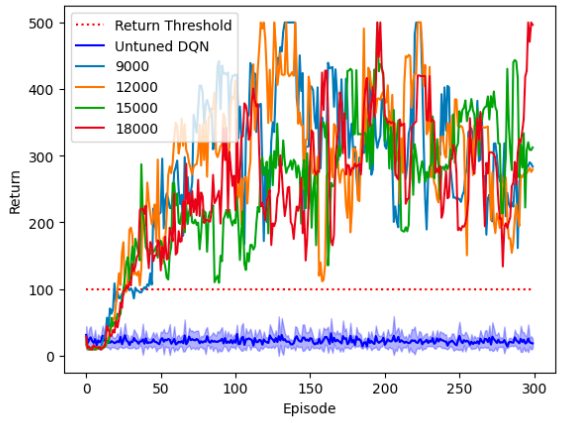


Figure 2: Effect of varying replay buffer capacity

In both Figures 1 and 2, the batch size and replay buffer were varied respectively with respect to the final chosen hyperparameters. However, the effects are difficult to distinguish in either graph with high levels of variance/noise due to an indeterministic training process. While the graphs provided little insight, the more objective evaluation metric of the sum of total rewards over each run helped choose the final hyperparameters

2.2 Learning Curve

Using the hyperparameters shown in Table 2, the learning curve of the final DQN can be drawn. Since the training is not a deterministic process, the results from 10 runs were averaged to create the learning curve shown in Figure 3.

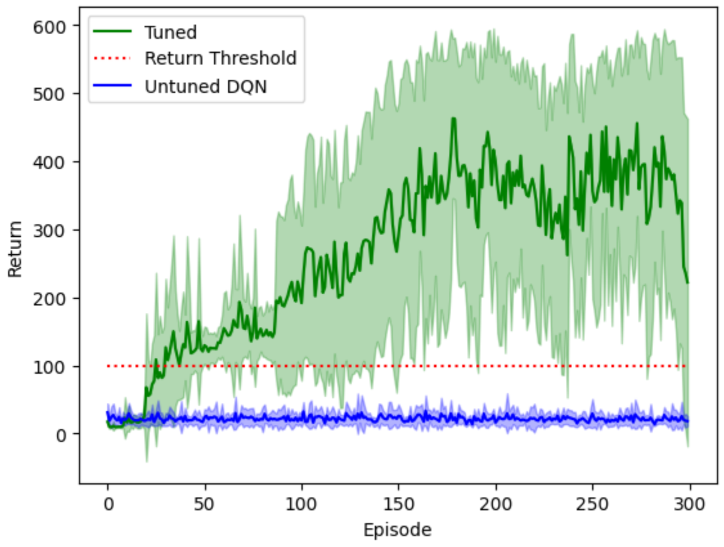


Figure 3: Average Learning Curve of Final DQN (10 runs)

As seen in Figure 3, the model was able to learn quickly and achieve a reward of 100 at approximately 25 episodes. Past 30 episodes, model performance remained above the return threshold until the end of the run, satisfying the performance requirement of 50 consecutive episodes of 100 rewards. Overall, performance continued to improve until approximately the 200-episode mark, from which performance started to plateau. While this is the general trend, model performance consistently had high-frequency variance throughout the training process due to sensitivity to initial weights resulting in an indeterministic process.

3 Visualising the DQN Policy

3.1 Slices of the Greedy Policy Action

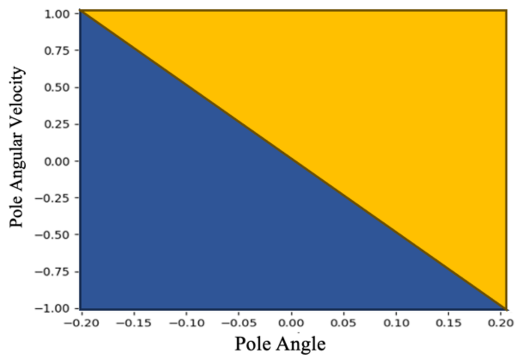
To understand slices of the greedy policy from the final DQN, an intuition can be made about what the slices should look like. First, the combination of angle and angular velocity can be used to understand what is happening to the pole-cart (situation). Subsequently, intuitive solutions can be derived from the cart-pole's situation for the simplest case when cart velocity is 0.

Table 3:Pole-cart Situation and Intuitive Solution given the Angle and Angular Velocity

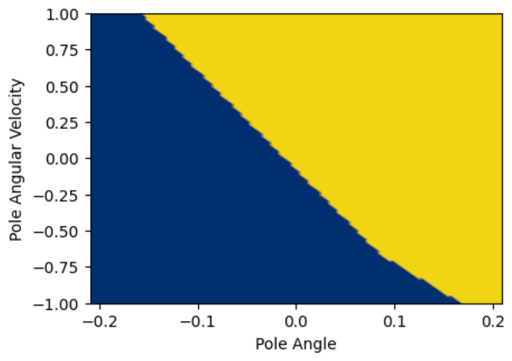
Angle	Angular Velocity	Situation	Intuitive Solution
Positive	Positive	Pole is falling right	Impart leftward force
Positive	Negative	Pole is rectifying from the left	Depends on the magnitude of angle and angular velocity
Negative	Positive	Pole is rectifying from the right	
Negative	Negative	Pole is falling left	Impart rightward force

In situations where the pole is being rectified, a more complex analysis is required. If a pole is rectifying quickly but is close to vertical (high angular velocity but low angle), it makes intuitive sense that a force should be applied in the same direction as its rectification. Conversely, for poles that are rectifying slowly and are far from vertical, a force should be applied in the opposite direction as its rectification. Combined with the intuitive solutions derived in Table 3, this knowledge can be used to give a general expectation of the model's greedy policy as shown in Figure 4.a. This theoretical expectation can then be compared against the actual slice of the model output when cart velocity is 0 (Figure 4.b). In the plots below, blue indicates leftward force and yellow indicates rightward force.

Note that the general pattern in Figure 4.a has been intuitively derived with some uncertainty. While it is certain that the upper right and lower left quadrants should be yellow and blue respectively, the slope and placement of the diagonal line that divides them is only generally known.



(a) Theoretical expectation (general pattern)

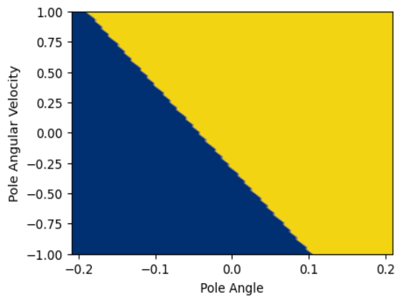


(b) Actual slice of greedy policy (cart velocity = 0)

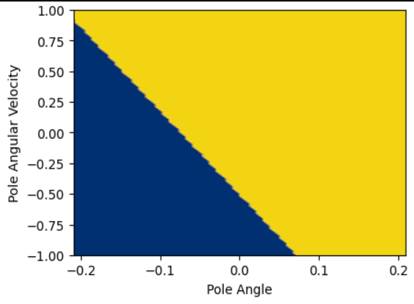
Figure 4: Expected general pattern of greedy policy slice at cart velocity = 0

As seen in Figure 4, the actual slice closely resembles the theoretical optimal greedy policy, just having a slightly steeper dividing diagonal between the blue and yellow regions. This supports that the model is learning correctly and is able to surpass the reward threshold.

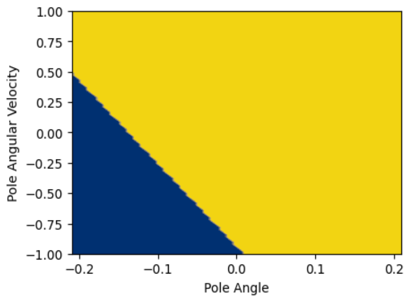
Continuing from the theoretical solution in Figure 4.a, increasing the cart velocity rightward should result in more leftward force actions (more yellow regions). As cart velocity increases rightward, poles in any situation are more likely to fall (and fail) leftward than in the cart velocity = 0 scenario. Hence, some situations that previously resulted in a leftward force greedy policy (blue) should be replaced with a rightward force (yellow) once cart velocity becomes significantly high. This is observed in the actual slices of the greedy policy when cart velocity is 0.5, 1, and 2, (shown in Figure 5)



(a) Cart velocity = 0.5



(b) Cart velocity = 1



(c) Cart velocity = 2

Figure 5: Slices of the greedy policy for varying cart velocities (cart position = 0)

3.2 Slices of the Q-Function

Similar to Section 2.2, the greedy Q-values can be shown via a 2D slice displaying the pole angle and angular velocity. As before, the cart position is fixed at 0 and four plots of differing cart velocities are given (0, 0.5, 1, and 2).

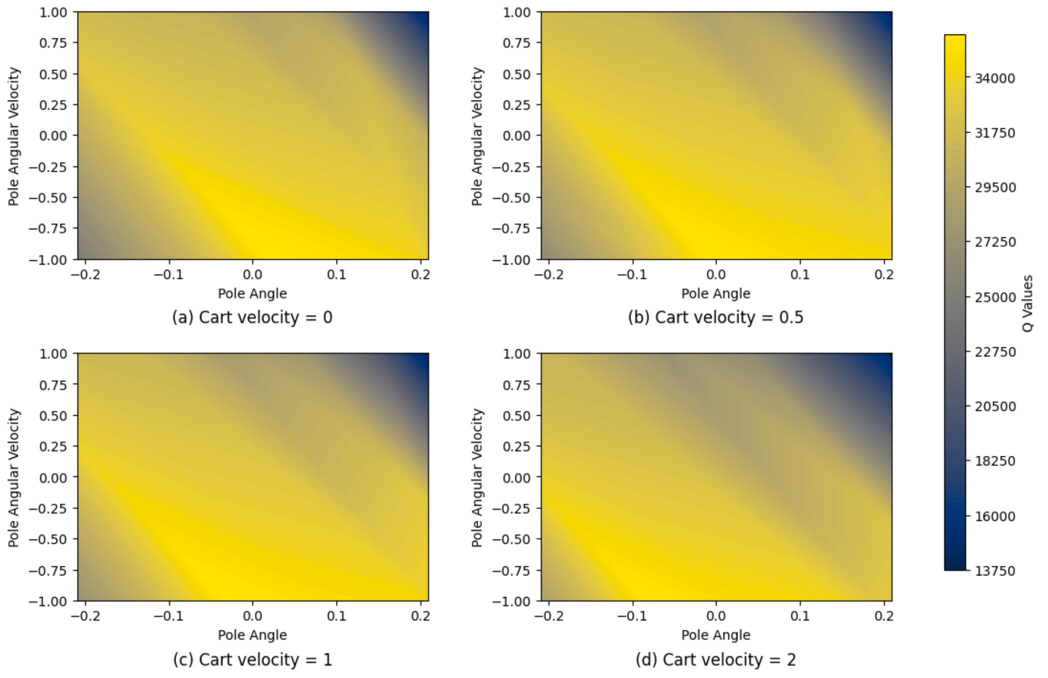


Figure 6: Slices of greedy Q-values for varying cart velocities (cart position = 0)

From Figure 6, a few patterns can be observed:

1. The upper right and lower left regions (falling pole situations) have the lowest Q-values.
2. Central regions (mostly upright poles with low angular velocity) and the upper left and lower right regions (rectifying pole situations) have the highest Q-values.
3. A single-direction diagonal colour gradient is formed, starting with yellow the in centre and darkening in the upper right and lower left directions.
4. As cart velocity increases, the upper right region darkens while the lower left region lightens.

Theoretically, falling pole situations should have the lowest Q-values as they are more likely to fail, as observed in Figure 6. As the pole angle and angular velocity increase in magnitude, the risk of failure also increases as the pole is closer to (and is quickly approaching) a terminal state. This increased risk is shown through the darkest shades seen in the upper right and lower left corners, supporting that the model is learning correctly. However, when cart velocity is 0, the lower left corner is unexpectedly less dark than in the upper right, implying that the model hasn't captured the symmetry of the cart-pole problem.

Conversely, situations where the pole is slow and mostly upright are the safest, as shown by the bright yellow regions in the centre. These regions of high Q-values also extend to the upper left and lower right regions, where the pole angle may be large but is rectifying quickly. In both scenarios, the risk of failure is theoretically low, therefore in alignment with the observed colour patterns in Figure 6.

Combining the patterns of high and low Q-value regions, diagonal bands of colour observed. This implies that moving within a same-colour band does not affect the risk of failure. Intuitively, this can be explained as moving closer to a terminal state (increase in angle magnitude) but being less likely to transition to that terminal state (increase rectification speed) does not affect the risk of failure.

In theory, an increase cart velocity (rightward) would result in lower Q values for a pole falling to the right and higher Q-values for poles falling to the left, as observed in Figure 6. This is explained by the rightward cart velocity requiring a leftward force to stop the cart from changing its position beyond 2.4 m, a terminal state. While this would rectify a left-falling pole (beneficial), it would cause a right-falling pole to fall faster (detrimental). Hence, left-falling poles (lower left region) are seen as more favourable states while right-falling poles (upper right region) are seen as less-favourable with higher risks of failure.