
REINFORCEMENT LEARNING

-

COURSEWORK 2 CART POLE ENVIRONMENT

Name: Jules Viard

Email: jcv23@ic.ac.uk

CID: 02461091

Department: Department of Computing

Course: MSc Computing (specialism Management and Finance)

Question I: Tuning the DQN.

In this section, we delve into the fine-tuning of a DQN model in the Cart Pole environment provided by OpenAI Gym. The objective is to establish an agent capable of achieving an average reward of 100 across 10 training runs, for over 50 episodes.

I.1. Hyperparameters.

To investigate the hyperparameter optimization, we mixed manual hyperparameter search with theoretical and practical considerations.

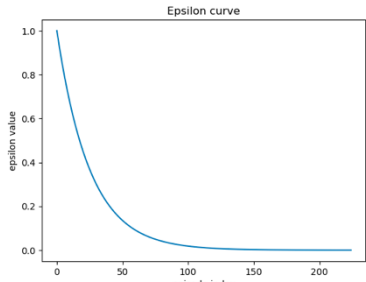
Our methodology for investigating hyperparameters encompassed the following steps:

- Step 1: Choosing a starting value thanks to theory, literature and practical considerations (especially from Q-Learning part of the previous coursework).
- Step 2: Designing narrow range of similar, potentially promising values.
- Step 3: Set up trials within this interval to identify the most suitable parameters for our agent model, guided by empirical observations.

In ‘Step 3’, we used four main criteria to assess and compare the hyperparameter choices:

- Average reward over episodes, and plateau reward at the end of the training.
- Variability and standard deviation.
- Number of episodes required to start learning and to reach satisfactory result (a reward of 100).
- Computational time.

It should be noted that the comparison of models under different hyperparameters depends on the overall configuration. Consequently, our conclusions regarding the selection of a particular hyperparameter are influenced by the other parameters in the configuration.

Hyperparameter	Value	Justification
Epsilon	Epsilon start = 1 Epsilon decay formula: $e^{-\frac{\text{index_episode}}{25}}$  <p>The graph, titled 'Epsilon curve', plots 'epsilon value' on the y-axis (ranging from 0.0 to 1.0) against 'episode index' on the x-axis (ranging from 0 to 200). A blue curve starts at (0, 1.0) and decays exponentially, reaching approximately 0.1 at episode 100 and approaching 0.0 by episode 200.</p>	In our epsilon-greedy algorithm setup, opting for a decaying epsilon facilitates an exploratory behaviour, initially characterized by random actions (with epsilon close to 1). As the number of episodes increases, the epsilon value decreases, reflecting a shift in strategy. Empirically, we obtained much higher average reward for epsilon decay strategy. The chosen formula aims to have roughly 30% of exploratory behaviour and 70% of exploitation.

This approach allows an initial exploration phase, where the agent learns from many examples, followed by an exploitation phase, which leverages the agent's acquired knowledge. Thanks to this efficient type of strategy in DQN model [1], the exploitation phase leads to better average reward (also experimented in the previous coursework).

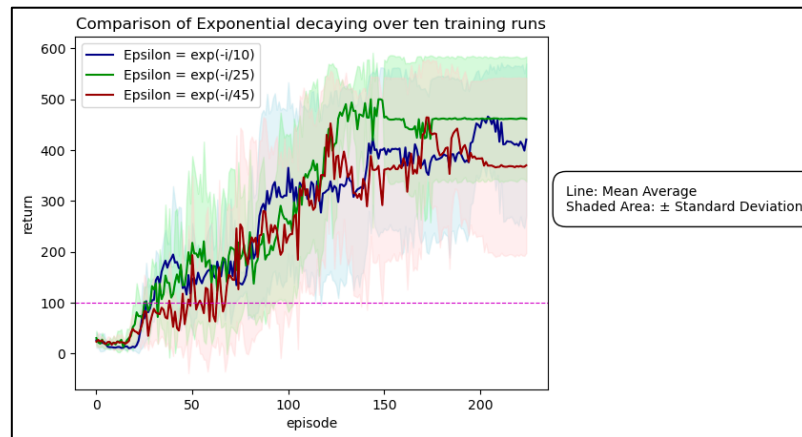


Figure 1: Comparison of exponential decaying impact on learning curve over ten trainings runs for each configuration.

As depicted in Figure 1, we obtained the best model (higher average reward and stability) with the epsilon decay formula stated in the tab. We especially highlight the plateau at the end of the training process: a feature absents in models using other epsilon decay formulas.

Hyperparameter	Value	Justification
Frequency of Update	Freq_update = 1	We decided to update the target network at the end of each episode. Consequently, we avoid an unstable training that can appear, for instance, when the network target is updated at every step (within an episode). Empirically, we obtained better performance with an update every episode, as each experience carries significant information.

Hyperparameter	Value	Justification
Structure of the neural network	Structure of the policy network: [4, 64, 64, 2] Structure of the target network: [4, 64, 64, 2]	The input layer matches the Cart Pole environment input space (4), and the output space, corresponding to the number of possible action (2). Given the small size of the input and output, we determined that two hidden layers, each with 64 neurons, offer an optimal balance. As our model is relatively small, we opted for fully connected layer to learn all possible combination from the previous layers.

Hyperparameter	Value	Justification
Buffer Size	Buffer size = 10 000	As every episode carries significant information, we opted for a large buffer. The large buffer size helps reduce correlation in training, allowing more efficient data utilization and rendering each episode important.

Hyperparameter	Value	Justification
Batch size	Batch size = 256	This parameter is linked to the buffer size and should be small enough compared to it. At the same time, it should be large enough to ensure generalization ability across different states. Based on our empirical findings, we selected a batch size of 256.

This approach was initiated after noting instances of 'catastrophic forgetting.' To address this, we examined several combinations of buffer and batch sizes, as illustrated in Figure 2:

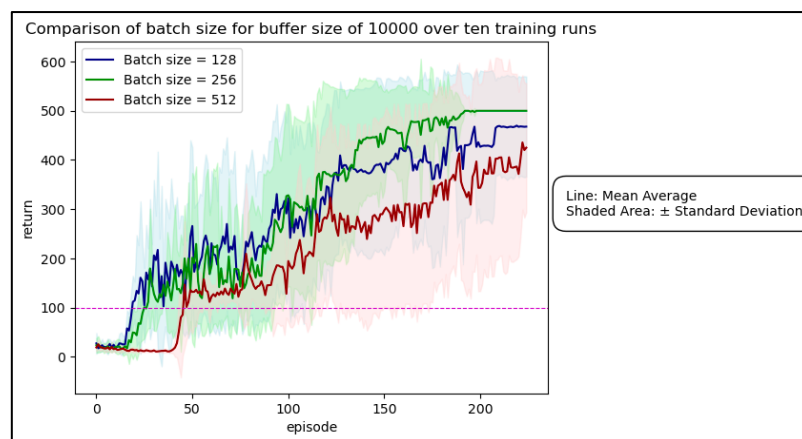


Figure 2: Comparison of batch size impact for a buffer size of 10000 over ten training runs for each configuration.

We observe, in Figure 2, that the batch size is correlated to a delay in the start of the learning.

Hyperparameter	Value	Justification
Optimiser	AdamW	Initially we used an Adam optimiser to have an interesting handling of the learning rate. In fact, this algorithm adapts the step size based on the first and second moment. With AdamW, we use a decoupled version [2] of Adam. This modification allowed us to independently set the 'weight decay' value, separate from the learning rate adjustments. Particularly we can choose an interesting regularization to handle the issue of overfitting and underfitting.

Upon detecting underfitting behaviour, we investigated the regularization. This setting was chosen as a trade-off between regularization and generalization capabilities. We aimed to avoid underfitting, which is associated with a lower average reward and is highlighted in blue in Figure 3.

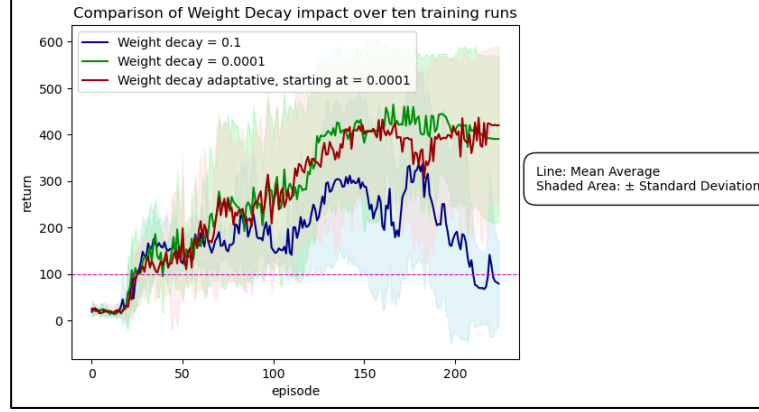
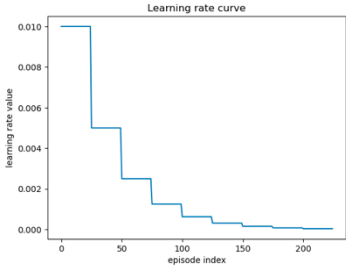


Figure 3: Comparison of weight decay parameter in the AdamW optimiser over ten training runs.

We also experimented with a dynamic 'weight decay' approach, indicated in red in Figure 3. In this case, we multiplied by 10 the 'weight decay' every 60 episodes. Finally, we did not retain this option. However, it can be interesting path for enhancing stability throughout the training process by incrementally increasing regularization, thereby reducing the risk of overfitting. Additionally, we used the AMSGrad variant of the AdamW algorithm [3]. Empirically, we observed better results on average with this variant.

Hyperparameter	Value	Justification
Learning rate	$lr_n = lr_0 * 0.5^n$ with $lr_0 = 0.01$ $n = \lfloor index_{episode}/25 \rfloor$ 'n' represents the number of times 25 episodes have passed. 	Even if AdamW algorithm dynamically adjusts the learning rate, we opted for a decaying learning rate. As it can be a powerful method [4] (and experienced from the coursework 1 for Q-Learning), we integrated a learning rate decaying every 25 episodes. Therefore, we have 9 phases with different learning rate caps. AdamW utilizes the provided learning rate as the upper limit for its own adaptation. By decaying the learning rate, we establish a global monitoring over AdamW. Thus, as the training progress, we enforce different phase of learning, encouraging stability and convergence.

We identified that an interesting interval for learning rate is $[10e-2, 10e-4]$. We chose this interval because the DQN agent is quite sensitive; a learning rate within this interval reduces oscillations, risk of divergence, and improves the average mean reward. Our design for the decay multiplier and starting value aims to maintain roughly the learning rate within this defined interval.

As illustrated in Figure 4, a starting value at 0.01 leads to a favourable outcome in terms of average reward, stability, and the number of episodes needed for achieving satisfactory results.

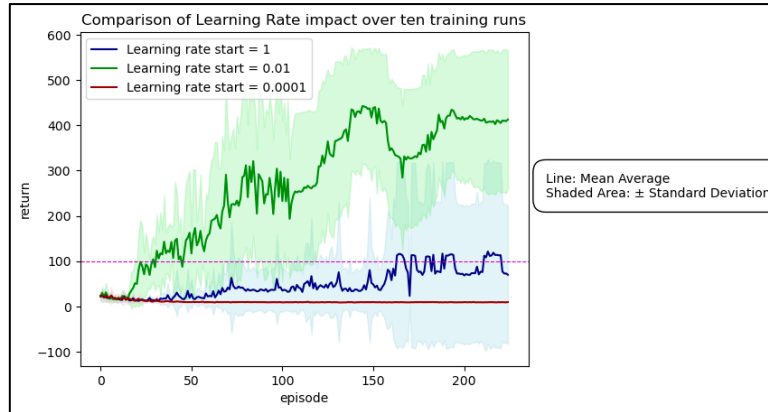


Figure 4: Comparison of performance for different starting leaning rate over ten training runs.

Hyperparameter	Value	Justification
Number of episodes	225	Upon examining the overall behaviour of our model, we concluded that the agent is relatively stable in the interval of 175 to 225 episodes. Based on this empirical observation across our analysis, we chose 225 number of episodes.

It should be noted that, given our decaying parameters (epsilon and learning rate), extending beyond 225 episodes offers no additional benefits. In fact, we calibrated the decaying factors, to achieve a roughly stability in the interval [175, 225].

I.2. Learning Curve.

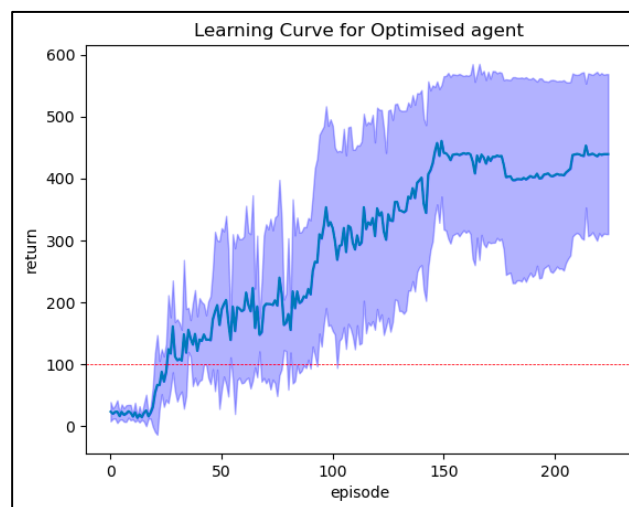


Figure 5: The learning curve of the optimised DQN model over ten training runs.

We can identify and correlate the behaviour of the learning curve (Figure 5) with our hyperparameter configuration.

Initially, there is a phase characterized by a relatively constant reward, related to our selected batch size and an epsilon close to 1.

Then, there is a discernible period of ‘exploration’ and ‘learning’, from episode 20 to 80, as evidenced by the variability and the gradual ascent of the average reward. In this phase, agent has large value of epsilon and learning rate. Consequently, the agent is more likely to not take the best action (epsilon) and still learn from this experience (learning rate). This phase of intensive learning is essential for establishing foundational knowledge of state-action pairs.

From the episode 80 to 150, we note that the agent’s average reward significantly rises. The 80th episode coincides with agent’s transition from an exploratory phase to an exploitation phase, characterized by decreasing values of epsilon and learning rate. In this phase, the agent is leveraging its learned experiences, prioritizing the best-known actions and giving less importance to new experiences.

In the final segment of the curve, referred to as the ‘plateau’ phase, is characterized by stability and high average reward. This is the period where the agent operates with its knowledge and refined strategy.

We observe a relatively significant variance. This can be attributed to the stochastic nature of the process and our hyperparameter choices (especially the weight decay) rendering the DQN model sensitive. However, the lower bound of the curve (mean minus standard deviation) consistently exceeds a 100-reward threshold over 50 episodes.

We compute the ‘plateau’ result by taking the reward during the final phase from 160 to 225. As the ‘plateau’ contains two main average rewards we derive the following formula:

$$plateau_{reward} = \frac{(225 - 208) * 440 + (178 - 160) * 440 + (208 - 178) * 400}{225 - 160} = 421$$

Question II: Visualise DQN policy.

In this section, we explore a specific ‘slice’ of the environment, focusing on a single agent when the cart pole is at the centre of the track. It should be noted that the slice observed in the following part depends on the agent’s training. Given the system's high variability, a different training run will induce different slices and other potential interpretation.

The velocities evaluated include [0, 0.5, 1, 2] (m/s).

II.1. Slices of the greedy policy action.

In Figure 6, we illustrate policy changes. These slices illustrate the agent's selected action (pushing right or left on the cart) based on the pole's angle and angular velocity.

We observe a roughly linear separation between the blue and yellow parts. This boundary represents the ‘stability’ line, where the agent maintains the pole. However, the agent's decision to act in either the blue or yellow region induces this idea of instability. Based on this separation line, we can observe that the agent has learned to discern the necessary action for different states of the system.

Moreover, the gradient of the ‘linear’ separation is negative, suggesting that the agent's strategy is to counterbalance the pole's direction of fall. For instance, if the pole falls with a positive angle (trigonometric convention), the dominant action is “push right on the cart” (action 1, yellow part) to prevent the pole falling, respecting fundamental physics of the cart pole system. The same phenomenon occurs when the pole falls with a negative angle: the agent has learned to push on the cart's left side (depending on the angle's and velocity).

The gradient of the separation is an indication of sensitivity and ‘aggressivity’ of the agent. In our case, we have a quite high gradient, meaning that our hyperparameter configuration leads to a sensitive agent.

As velocity increases, we observe a downward shift in this linear separation. The policy progressively (as velocity increases) favours action 1, especially when the pole falls with a positive angle. This coincides with the observation that for a larger range of angle, action 1 predominates. The higher the velocity, the higher the angular velocity applied to compensate for the falling. This observation confirms the agent's understanding of decision-making during high-velocity movement.

For the zero-velocity case, a non-zero angular velocity is noted even at a zero angle. This implies that our agent does not completely seek for stability and opts for ‘aggressive action’, resulting in the pole's instability in a neutral position. However, the agent does not make necessarily the pole falls.

Overall, the investigation of slice of greedy policy suggests a potential overfitting behaviour (high sensitivity). To counteract overfitting and improve generalisability, we can refine our decaying hyperparameter. A promising approach involves using smaller decaying factors, implementing stronger regularization, and extending the training period.

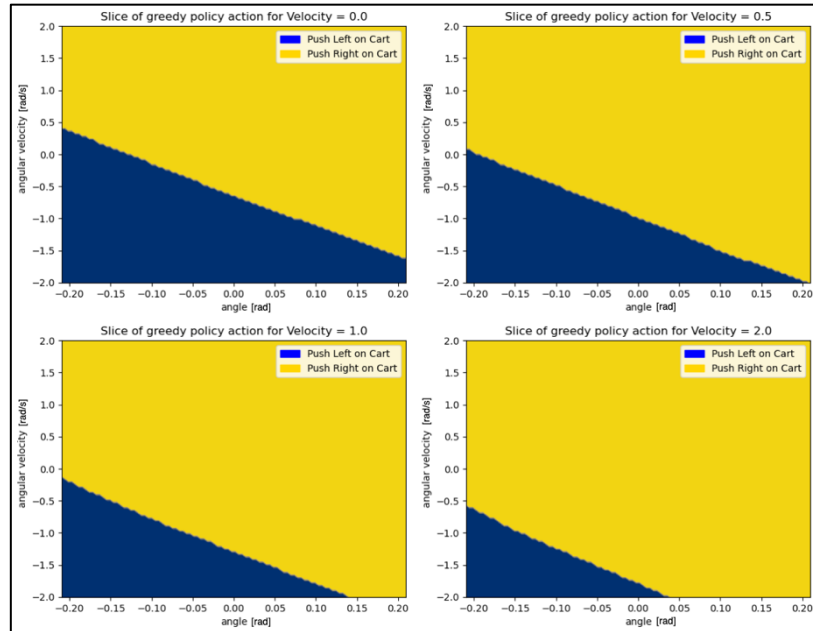


Figure 6: Slice of the greedy policy action for four values of cart's velocity.

II.2. Slices of the Q function.

In Figure 7, we have a visual interpretation of the Q-value for a given velocity, which offers insights into the 'benefit' of different states.

The yellow regions represent states where the Q-value are high, indicating a higher potential reward for actions taken within these zones. On the other hand, the blue regions indicate a lower reward, implying less favourable stability conditions for the pole. For instance, when the pole falls with a positive angle, actions that result in a negative angular velocity correlate with higher rewards.

Regarding the blue region, it can represent either actions that contribute to the pole's fall or unexperienced situations. Typically, actions that encourage the fall yield low rewards. For example, applying the action 0 when the pole falls with a positive angle would be a sub optimal decision.

We can identify the separation lines, illustrated in Figure 6, with the yellow diagonal region (from upper left to bottom right) in each plot of figure 7. This region's negative slope aligns with the interpretations in section II)1). In fact, it traduces the agent's strategy of rewarding the action that counterbalance the pole's angle to reach a 'stability' state. In other words, the angular velocity is 'opposite' to the pole's angle. So, the diagonal region represents the most desirable state and contribute to the pole's stabilization at the center.

However, due to the model's sensitivity and potential overfitting, this dynamic towards stability is not completely achieved (as described in II)1)).

The same concept of downward shift, for the diagonal region, is evident in Figure 7. Indeed, as velocity increases, the shift of this region indicates a 'correction' of the strategy.

We also have a clue of overfitting issue due to the large blue region and the sparse yellow region. This means that the agent has not learned enough and tends to reward a broad range of action, leading to instability.

The distribution of Q-value in Figure 7 reflects the reward strategy towards prioritizing the pole's stabilization.

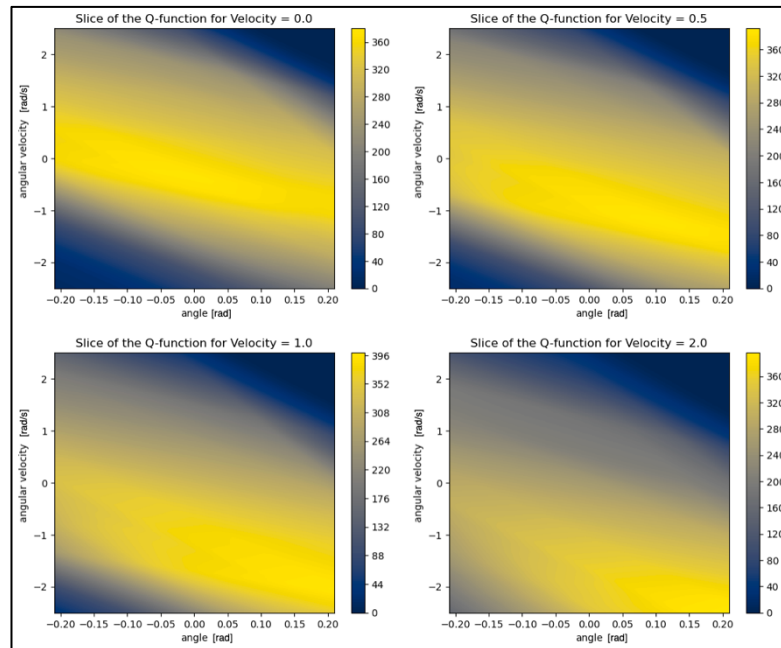


Figure 7: Slice of the Q-function for four different values of cart's velocity.

References

- [1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
- [2] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
- [3] Liu, L., Jiang, Y., He, P., Lv, Y., & Wang, Z. (2019). On the convergence of Adam and beyond. arXiv preprint arXiv:1904.09237.
- [4] Even-Dar, E., & Mansour, Y. (2003). Learning rates for Q-learning. *Journal of Machine Learning Research*, 5(1), 1-25.