ASSIGNMENT 2: Deep Q-learning, Q-learning and SARSA

The plots in the report have a mean from the previous hundred plotted in either orange or red depending on the task. The continuous plot colors are blue or green. When the numbers of episodes are changed when comparing plots the training is done again, so mind differences in the overlapping periods.

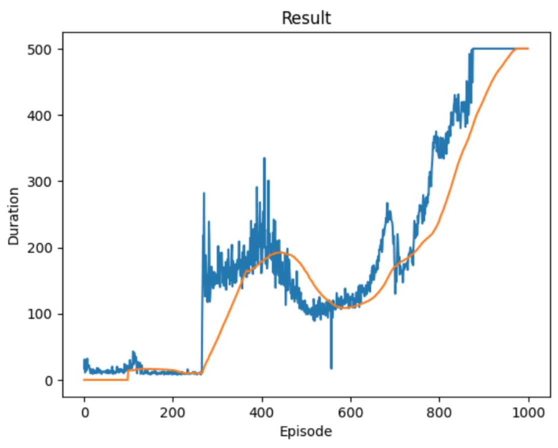
TASK 1

1. **Why do we need a policy net and target net?**

The policy network is the network with actions from the different states for making decisions based on the script's policy. The policy to be followed is to maximize the total reward (Doshi, 2021). The Q-value is deducted from the Q-network. Target network is a structural copy of the Q-network. The target network updates its weights (values) at a different pace than the policy and Q-network. The target net is updated “softly” to ensure more stable learning for differences in the target (moving end state or target depending on the action). By facilitating for a buffer on moving targets the risk of divergent learning is less. The target network periodically adjusts the Q-network (the weights) to provide smooth learning (*Target-Network in Reinforcement-Learning*, n.d.).

In this specific case where the performance is very measurable, but the state is constantly changing and the target is moving as the cart is moving, policy networks and target networks are useful. The deep Q-learning benefits from having a stable target, else the Q-network would not be able to match predicted values to actual values due to constant change. This would prevent the model from learning (*Target-Network in Reinforcement-Learning*, n.d.).

2. **When do you think a model converges?**

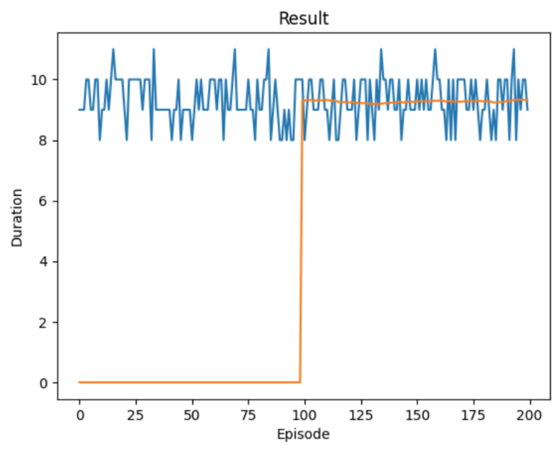
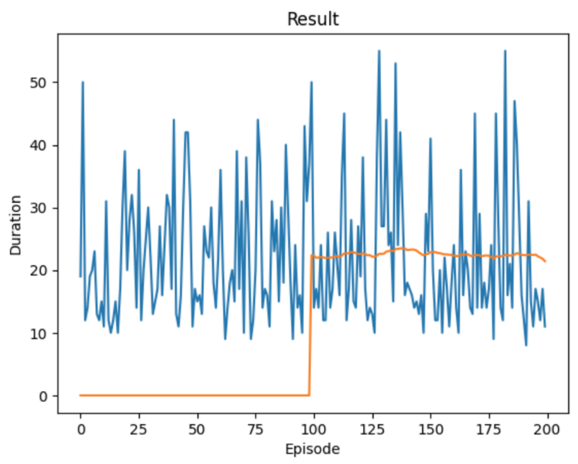


Here is the plot for 1000 episodes, with no other changes.

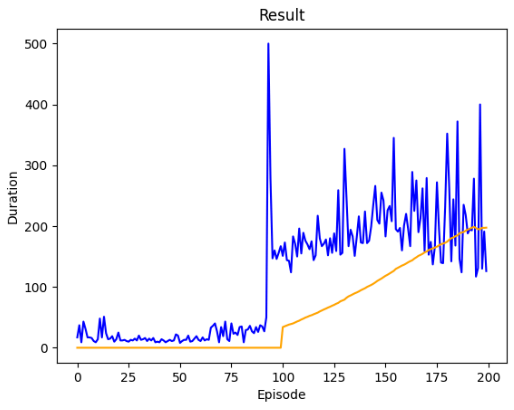
From the plot above one can see that the last 50-100 episodes all score the same. Before this the scores are fluctuating between the whole specter of scores. The conclusion is thus that the unchanged model converges when around 900 episodes have passed. This will depend on exploration probability, learning rate and discount factor along with duration of each episode, but for the provided model the convergence starts at around 950 episodes. Maybe a higher ceiling for duration would prove that the model converges later.

3. **Observe the effects of changing the exploration vs exploitation. (Show results in plots or print statements)**

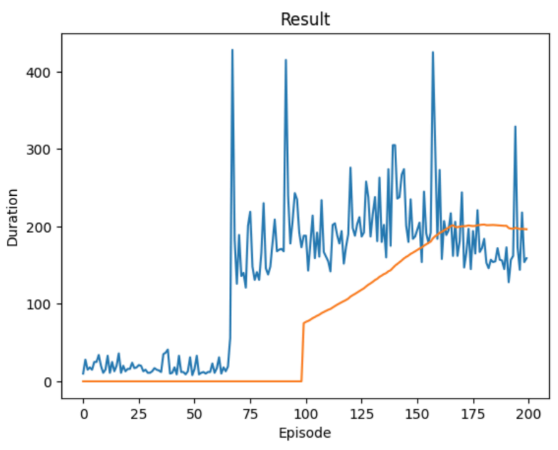
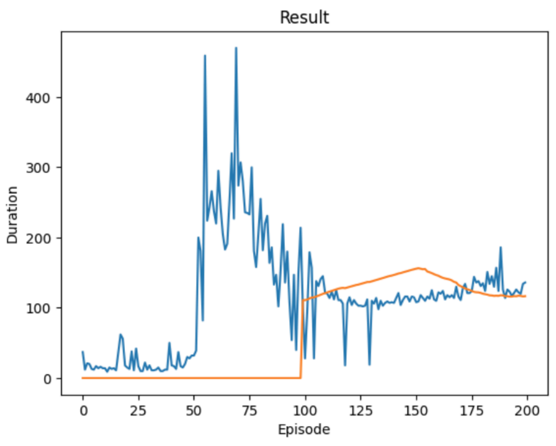
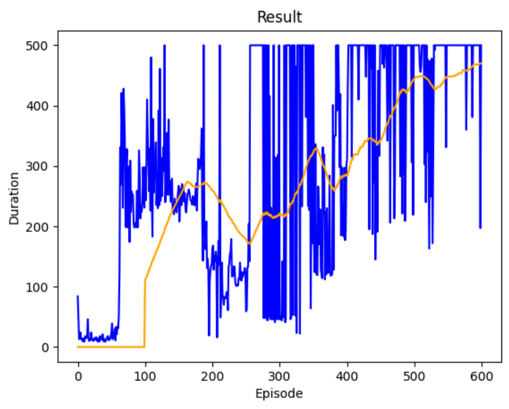
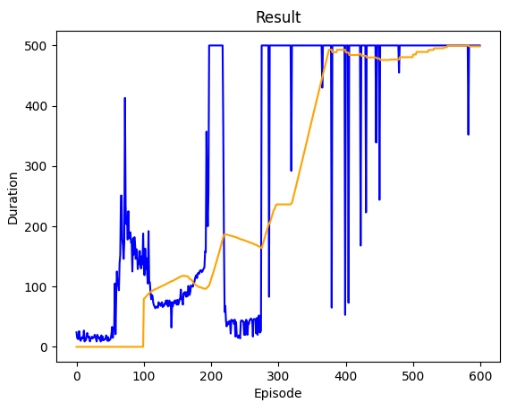
This is done by altering the threshold in the select_action function, either doing the highest performing action every time or exploring a random action for the state every time. Exploitation is doing the action with the highest Q value, but exploration is expanding the knowledge of the space to improve performance in the long run.

Complete 	Complete 
Exploitation	*Exploration*

A combination of exploring and exploiting knowledge of the action space is the best recipe for achieving a high score, as illustrated below. If nothing is explored then the maximum reward is within the immediate vicinity of the starting state. Else, if only exploration is done the continuation of a good action route will be disregarded to explore a worse route. Exploring many possibilities to gain as much information as possible, then choosing fitting actions based on policy or maximum reward is beneficial.

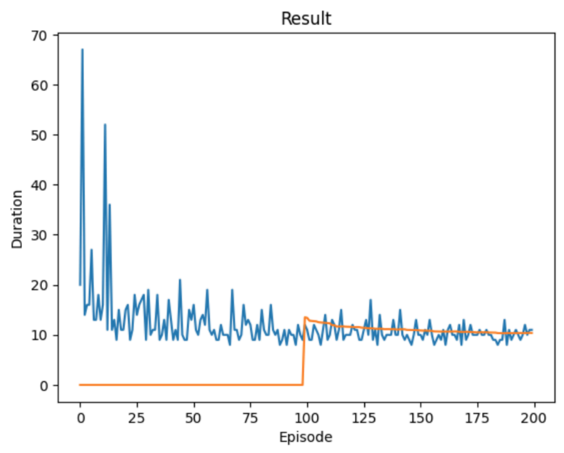
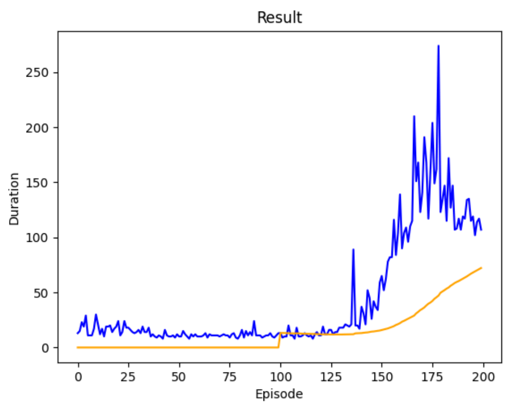
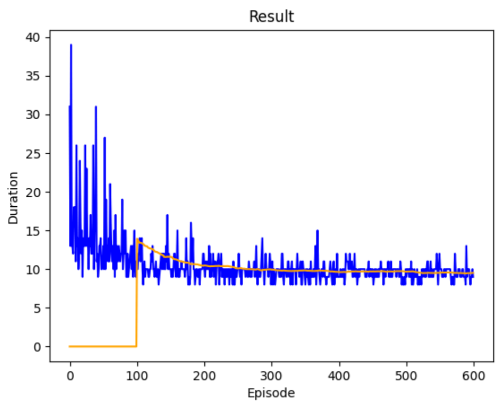
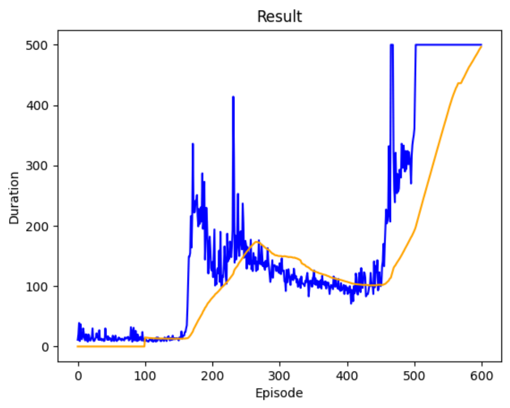

3 layers 200 episodes, normal exploitation and exploration

4. **Vary the number of layers in the model and plot episode vs duration plots. Maybe good to keep an eye on how long!**

	
	
4 layers	*5 layers*

After 150 episodes both models are very similar, but afterwards the five-layered model sinks in performance and the four-layered continues to improve a bit and then has stable learning when approaching 200 episodes. It seems as if the model is worse performing when having five than four layers just before 200 episodes. When moving into the later stages of training the fifth layer may complement the model in a way the original four cannot, thus making the model perform better at the end.

5. Implement without the replay buffer and observe performance.

`Complete` 	
	
Replay memory is 0	*Replay memory is 10000*

Having no past experience to train on the model becomes redundant and non-learning. This is the purpose of cashing past states and rewards, to build a learning model. One can observe that with only 200 episodes the replay buffer is very important to the learning of the model. After 600 episodes with the replay memory the model performs perfectly.

TASK 2

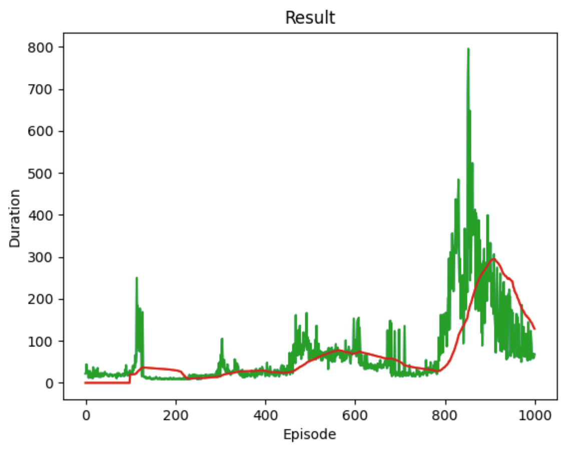
1. Why do we need a policy net and target net?

The policy network is the network with actions from the different states for making decisions based on the script's policy. The policy to be followed is to maximize the total reward (Doshi, 2021). The Q-value is deducted from the Q-network. Target network is a structural copy of the Q-network. The target network updates its weights (values) at a different pace than the policy and Q-network. The target net is updated “softly” to ensure more stable learning for differences in the target (moving end state or target depending on the action). By facilitating for a buffer on moving targets the risk of divergent learning is less. The target network periodically adjusts the Q-network (the weights) to provide smooth learning (*Target-Network in Reinforcement-Learning*, n.d.).

In this specific case where the performance is very measurable, but the state is constantly changing and the target is moving as the cart is moving, policy networks and target networks are useful. The deep Q-learning benefits from having a stable target, else the Q-network would not be able to match predicted values to actual values due to constant change. This would prevent the model from learning (*Target-Network in Reinforcement-Learning*, n.d.).

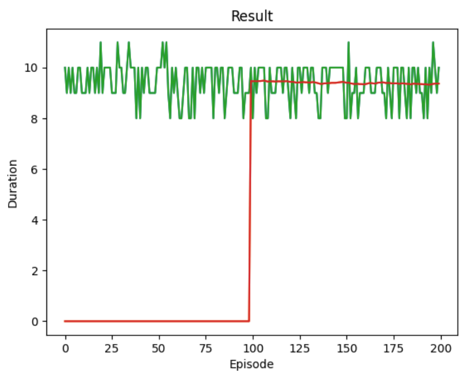
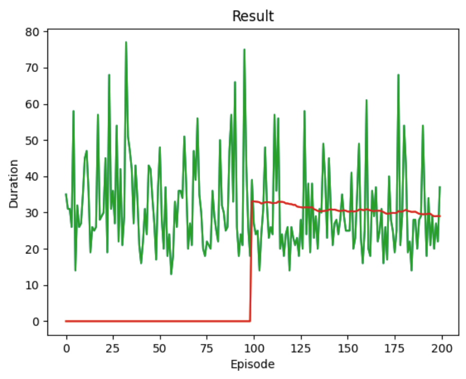
As the target in the 2D version also can move in more directions I think the target network is especially important. This is because I think the complexity if the state is much bigger, thus leaving the target Q-value and the predicted value to differ more in the 2D version. We need target and policy networks in both environments, but i think they play a bigger part in task 2.

2. **When do you think a model converges?**


3 layers 1000 episodes

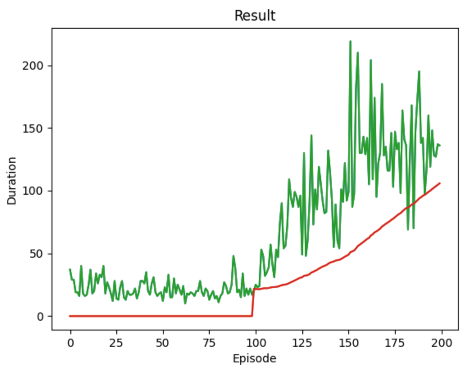
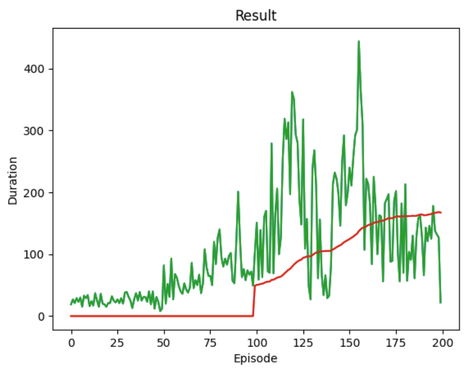
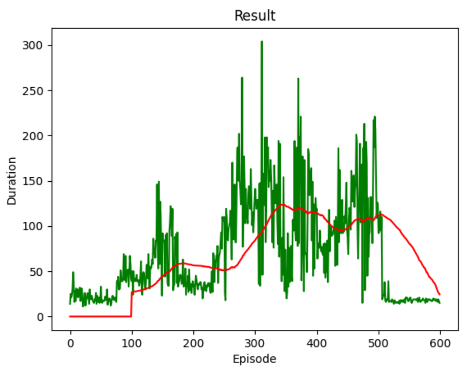
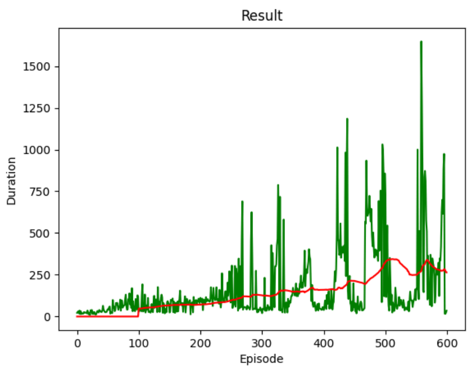
With the original setup the model has no converging curve up to 1000 episodes. There are stable parts of the model, but none are convergent. This may suggest that 3 layers are not sufficient to make a stable model for the environment. Or, it may crave more than 1000 episodes to converge with 3 layers. The models with 4 and 5 layers are not convergent either, though running only 600 episodes. These images are the two latter in task 2.4.

3. **Observe the effects of changing the exploration vs exploitation. (Show results in plots or print statements)**

	
Only regarding policy in select action, 3 layers. Exploitation	*3 layers and only random samples in select action. Exploration*

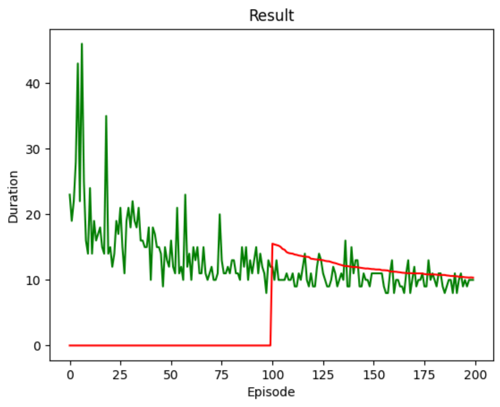
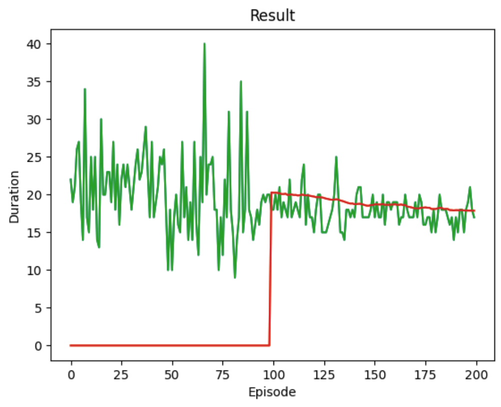
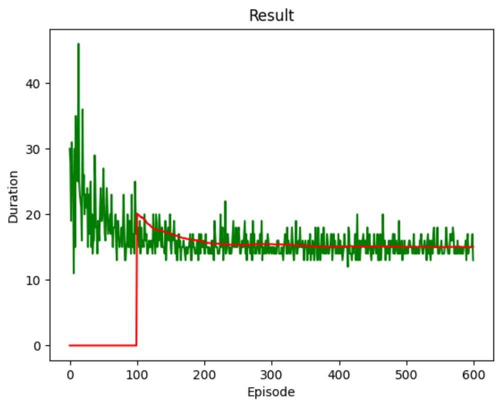
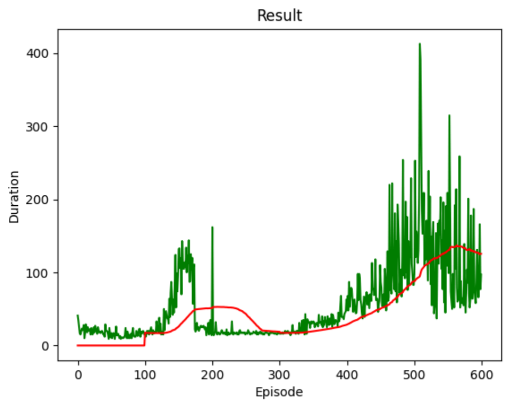
The result is the same as from task 1, as expected since the total exploitation and exploration is deterministic for both models. If there was just an adjustment to the sensitivity of exploration, either less or increased the model would be different. Low exploration yields slow, stable, learning and high exploration yields random jumps in performance, but would probably have the model with the highest maximum.

4. **Vary the number of layers in the model and plot episode vs duration plots. Maybe good to keep an eye on how long!**

	
	
4 layers in the 2d version	*5 layers in the 2d version*

In this task it seems the 5 layers are better performing than 4 layers. Both after 200 and 600 episodes the 5-layered model is better performing, and is more stable. The more complicated two-dimensional task may require an extra layer to account for the extra dimension, and the performance boost for the extra layer is greater in this environment than for the one-dimensional task. Though, the 5-layered model performed better in both cases, the difference is bigger in the two-dimensional environment.

5. Implement without the replay buffer and observe performance.

	
	
3 layers with 0 memory	3 layers with 10000 memory

The replay buffer is not that important in 200 episodes, but increases the average duration by roughly ten points. Both models are declining in performance up to 200 episodes. The difference the memory makes is more prevalent after 200 episodes, where the model without a memory buffer is converging to 15 and the model with memory peaks at around reward 130. One can clearly see improvement in the models performance with the ascending trajectory of both the mean and individual graphs.

TASK 3

When implementing the Q-learning and SARSA methods I have been inspired by articles, and the specific places where I have taken inspiration are marked in the scripts and have links to the sources in the code. I have taken inspiration in the set-up of the model from TowardsDataScience (Amine, 2020) and the precode. Though the example used in TDS's article is different and therefore there are substantial differences, such as discretization, in my implementation.

Technical background

The gymnasium environment is made up of a cart and a pole both moving in one dimension. The possible observations are therefore cart position and velocity and pole angle and velocity. The goal is to make actions to balance the pole in the cart for the longest possible time (*Gymnasium Documentation*, n.d.).

One important aspect is the discretization of the space - as the observation space is continuous, and needs to be discretized for the model to be able to learn. This idea is inspired by ((Slater, 2019) & (HLeb, 2019)). The basic thought is splitting the continuous observation space into smaller chunks, bins. Based on the continuous values from the state, the values within one bin are regarded as the same. An example would be to make 40 equispaced borders for "pole angles" between -20 and 20 degrees. Then a pole angle of -13.3 and -13.9 would be regarded as the same. Else the amount of different states would be infinitely big and the model would not be able to act.

The discretization of the observation space is the foundation for the Q-table, as the possible states are not set. The shape of the table (matrix) will be the length of each observation for each of the observations and the possible actions possible in these states. For an observation space like this instance, with four observations with length 100 each and two possible actions in each state would be (100, 100, 100, 100, 2) (*Gymnasium Documentation*, n.d.). Trial and error led to me choosing chunk size 100, which is not small nor very large.

```
N_ACTIONS = env.action_space.n
[...]
N_CHUNKS = 100
cart_pos = np.linspace(-4.8, 4.8, N_CHUNKS)
cart_vel = np.linspace(-20, 20, N_CHUNKS) # Actually -inf and inf
pole_ang = np.linspace(-0.418, 0.418, N_CHUNKS)
pole_vel = np.linspace(-20, 20, N_CHUNKS) # Actually -inf and inf
observation_space = [cart_pos, cart_vel, pole_ang, pole_vel]
Q_TABLE_SHAPE = cart_pos.shape[0], cart_vel.shape[0], pole_ang.shape[0], pole_vel.shape[0],
N_ACTIONS
q_table = np.zeros(Q_TABLE_SHAPE)
```

Q-learning and SARSA are forms for Reinforcement learning where the goal is to map out the observation and action space and to perform actions in line with the highest predicted cumulative reward. This predicted cumulative reward is formed in the Q-table is based upon the actions in each state (Somani, 2024). The difference between QL and SARSA is the way that they update the values in the states, and what actions they chose to perform (*SARSA Reinforcement Learning*, 2019).

Design and implementation

The design is inspired by TDS author Amine's guide for Q-learning in gym environments (Amine, 2020). My design is different to this due to the discretization of the observation space and thus the Q-tables unique shape. As mentioned the Q-table is made from the unique states and actions possible. Aswell I have also implemented SARSA, not just Q-learning.

When the empty Q-table is made the training can begin. The state is retrieved and an action is made. The action is either random or the one yielding the highest q-value (predicted cumulative reward) from the state. A declining exploration probability is deciding if the action is random or not, this is to utilize exploration and exploration. The action is made, and the new state is retrieved and discretized. This is where SARSA and Q-learning part ways briefly.

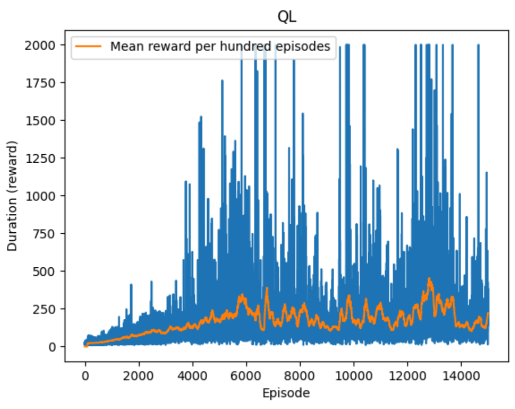
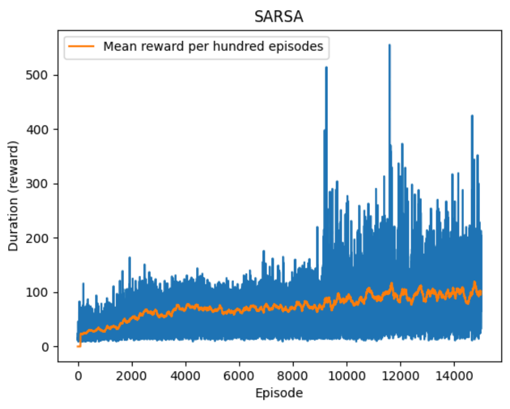
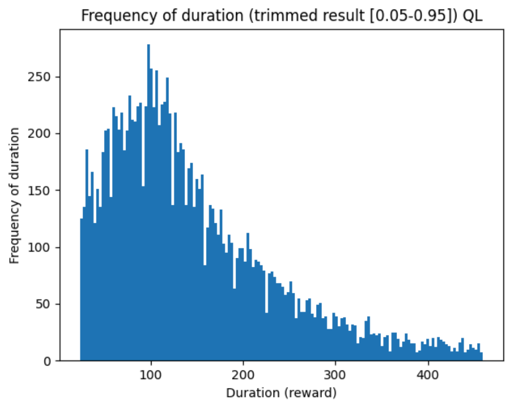
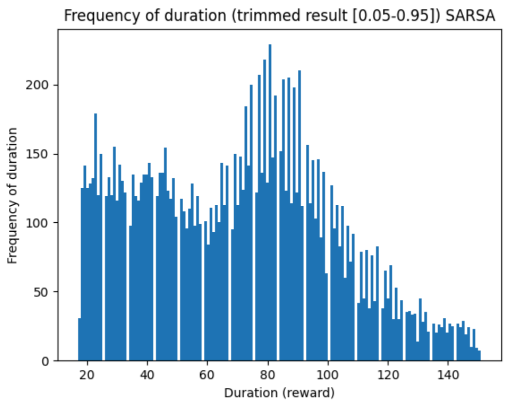
SARSA has an exploration probability to either make a random action or the action yielding the highest q-value next. The Q-learning model always acts in favor of the highest q-value here (Duong, 2021). The rest is similar for SARSA and QL. The chosen next action's q-value is then stored. The current states q-value is altered with respect to the new and current states q-values, with a learning rate and a discount factor for valuing the next states q-value ((clwainwright, 2019) & (PolBM, 2016)).

Then the reward is added to the round's total reward and a check is performed to see if the pole has fallen too far or if the round length is up. The next state becomes the current state, and if the check passes the loop runs again. For each iteration the exploration probability declines, aiming to explore in the start and exploit information toward the end.

```
EXPLORATION_PROB = 1
EXPLORATION_DECAY = 0.001
MIN_EXPLORATION_PROB = 0.01
DISCOUNT_FACTOR = 0.99
LEARNING_RATE = 0.125
N_EPISODES = 15000
MAX_EPISODE_LEN = 2000
```

The parameters used in both the SARSA and QL scripts are the same, and are in the table above. They are tested, but not to perfection. The rendering of the model is also removed to reduce time consumption when training. This has made it possible to have loads of long lasting episodes to train on. The discount factor is quite high since I want to value the reward in the next step also. The learning rate is surprisingly high, but it performs well and stably over the episodes, especially for SARSA.

Results

	
	
Q-Learning	*SARSA*

As one can see the Q-learning model performs better than the SARSA model. The Q-learning model always makes the action that yields the highest q value, but the SARSA model randomly makes other next actions as well as the highest q value. I think this may be the reason for the difference in the models. As one can see by the result over the bulk of thousand episodes below; the QL outperforms SARSA in every single one.

The results also show that the most frequent reward from Q-learning is bigger than and more frequent than the most frequent SARSA score. The top and bottom 5% are shaven off to get a more representative comparison, disregarding outliers from both ends. While the mean per hundred for SARSA never reaches over reward 100 the mean per hundred for Q-learning is regularly above 150.

Thousand mean step 1: 28.498	Thousand mean step 1: 27.736
Thousand mean step 2: 60.138	Thousand mean step 2: 41.137
Thousand mean step 3: 93.969	Thousand mean step 3: 59.752
Thousand mean step 4: 122.65	Thousand mean step 4: 64.252
Thousand mean step 5: 168.778	Thousand mean step 5: 71.403
Thousand mean step 6: 241.369	Thousand mean step 6: 65.577
Thousand mean step 7: 213.516	Thousand mean step 7: 69.652
Thousand mean step 8: 195.163	Thousand mean step 8: 70.774
Thousand mean step 9: 176.595	Thousand mean step 9: 74.262
Thousand mean step 10: 179.415	Thousand mean step 10: 82.269
Thousand mean step 11: 204.977	Thousand mean step 11: 89.075
Thousand mean step 12: 169.103	Thousand mean step 12: 94.395
Thousand mean step 13: 282.175	Thousand mean step 13: 95.105
Thousand mean step 14: 243.882	Thousand mean step 14: 96.106
Thousand mean step 15: 149.946	Thousand mean step 15: 98.42
QL	SARSA

Challenges

Some challenges I faced during the tasks were the understanding of how the Q-table should be defined for a continuous observation space. This required some research and led me to the solution I presented above. Aswell, the understanding of the difference between SARSA and Q-learning. The specification of the difference between QL and SARSA that enlightened me the most is from (*SARSA Reinforcement Learning*, 2019).

Some coding related challenges I faced was the tuning of the parameters. The tuning was done by trial and error, and some "guesstimtes" that seemed to work fine. And making sensible plots that illustrate the difference, as outliers may skew the image when frequency of durations.

Resources

Amine, A. (2020, December 19). *Q-Learning Algorithm: From Explanation to Implementation*. Medium. https://towardsdatascience.com/q-learning-algorithm-from-explanation-to-implementation-cdbeda2ea187

clwainwright. (2019, September 21). *Answer to 'Understanding the role of the discount factor in reinforcement learning'*. Cross Validated. https://stats.stackexchange.com/a/428157

Doshi, K. (2021, April 24). *Reinforcement Learning Explained Visually (Part 6): Policy Gradients, step-by-step*. Medium. https://towardsdatascience.com/reinforcement-learning-explained-visually-part-6-policy-gradients-step-by-step-f9f448e73754

Duong, V. H. T. (2021, February 23). *Intro to reinforcement learning: Temporal difference learning, SARSA vs. Q-learning*. Medium. https://towardsdatascience.com/intro-to-reinforcement-learning-temporal-difference-learning-sarsa-vs-q-learning-8b4184bb4978

Gymnasium Documentation. (n.d.). Retrieved 5 March 2024, from https://gymnasium.farama.org/environments/classic_control/cart_pole.html

HLeb. (2019, May 11). *Answer to 'Can Q-learning be used for continuous (state or action) spaces?'* Artificial Intelligence Stack Exchange. https://ai.stackexchange.com/a/12258

PolBM. (2016, June 30). *Answer to 'Understanding the role of the discount factor in reinforcement learning'*. Cross Validated. https://stats.stackexchange.com/a/221472

SARSA Reinforcement Learning. (2019, June 14). GeeksforGeeks. https://www.geeksforgeeks.org/sarsa-reinforcement-learning/

Slater, N. (2019, May 11). *Answer to 'Can Q-learning be used for continuous (state or action) spaces?'* Artificial Intelligence Stack Exchange. https://ai.stackexchange.com/a/12257

Somani, A. (2024, February 27). *Assignment 2: Reinforcement Learning*.

Target-network in reinforcement-learning. (n.d.). Retrieved 2 April 2024, from https://livebook.manning.com/concept/reinforcement-learning/target-network