Alvaro Prat Balasch

CID: 01066209

Reinforcement Learning Part 2: Coursework Part 1

Question 1(i): We can clearly observe improvements in training stability from online learning using a single step (Figure 1a) to mini-batch learning through an experience replay buffer (Figure 1b). In the former, the variance in training is too large for the Q-value to stabilise: learning from one single example is too stochastic, resulting in biased gradient descent and localisation (agent may move to a similar state with correlated values and old transitions can be "forgotten" in the Q-network). Note that the variance hardly reduces as the agent explores the environment and the average loss remains fairly constant. On the other hand, introducing an experience buffer stabilises the training as the update of the Q-network becomes more reflective upon the true values observed. Moreover, it allows training from previously observed transitions, increasing its generalisation towards the true Q-function.

Question 1(ii): Using mini-batch training (Figure 1b) significantly increases convergence and stability. However, back-propagating $N = 50$ samples is computationally more expensive (logarithmic). Regarding online training, we can consider the fact that training begins at the very first step, whilst during mini-batch learning we must wait for the Nth step. Nevertheless, mini-batch training is still fast considering the size of the Q-network and that we can vectorise the batch. In addition, the increased stability observed in mini-batch training outruns online learning. Hence, using an experience buffer may be regarded as more efficient in terms of improving the overall predictive accuracy in realistic timed conditions.

Question 2(i): The Q-values are displayed in Figure 2a. Here, we observe the bottom-left Q-values to be resolved whilst the top-right quadrant is not. This may be explained by the fact that we have short episodes of 20 steps, and our actions are random meaning that the expected states to be explored lie amongst the bottom-left quadrant. On the other hand, there is almost no exploration in the top-right quadrant, impeding the Q-value to be predicted with lower accuracy (no training data).

Question 2(ii): The greedy policy after training for 500 steps is displayed in Figure 2b. We can immediately observe that the agent gets stuck at Pixel $(500, 1100)$. The first steps of the greedy policy were correct, as the Q-network trained in that region exhaustively. Nevertheless, the Q-values are poorly defined further from the initial state. This impedes the agent to reach the goal state as it believes that the best action is to go right. However, the obstacle in the environment impedes it from doing so. This problem could potentially be adjusted with more training episodes, however another fix could be to penalise transitions towards a wall with negative rewards.

Question 3(i): We have now included the bellman equation with a discount factor γ of 0.9 and the MSE losses are displayed in Figure 3a. Even though the loss initially drops very fast, the variance in MSE leads to unstable training and divergence. The

Q-network seems to have lost its generalisation capabilities: when trained at one state, it is updated for similar states which may then change the prediction for the same state it is evaluating. This leads to instabilities and it is observed in this Figure as the loss seems to artificially decrease exponentially in some occasions.

Question 3(ii): When including a target network (Figure 3b) we observe how the sinusoidal shape of the loss is reduced significantly with respect to the bellman equation without a target network (Figure 3a). Moreover, the variance of the prediction is reduced. This is especially true in the last steps where the loss stabilises, allowing potential convergence. An increase in stability arises as the temporal difference contribution to the bellman equation is stabilised along each episode (updated every 20 steps). This inhibits the Q-network to bootstrap from itself allowing it to settle down before updating the target network (observed by the periodical spikes every 20 steps representing the target network update).

Question 4 (i): From Figure 4 we find an optimal δ of 4×10^{-3}. This gives a highly superior performance to a δ of 0. The reason for this is that we allow the agent to take weighted actions after learning some aspects of the environment. A δ of 4×10^{-3} is small enough for the agent to randomly explore, however it is more efficient as it allows weighted actions to take place after learning about the initial states (close to the red dot). This enables the agent to get further into the states that have not yet been explored as the expected state position translates towards the goal, whereas using $\delta = 0$ would not allow the agent to randomly reach the goal given 20 steps.

Question 4 (ii): A high value of epsilon decay $\delta = 1$ means that the agent can only randomly explore the environment once before greedy exploration. This is clearly an overkill: we cannot explore greedily in a world we do not understand as we potentially observe the same repeated states over and over (despite Q-values converge in greedy trace). This has catastrophic effects on the total rewards as shown in Figure 4.

Question 5 (i): My optimised reward function: $r = 0.6 - d^{1.4}$, where r is the reward and d is the distance to the goal, focuses increased Q-value gradients near the goal. Moreover, a penalisation of $r = -0.2d$ is applied when the agent does not move in the next state, allowing it to to escape local minimas when blocked within a wall, in contrast to using Eucledian distances only: Question 2(ii). Ultimately, an incremented reward of $r = 0.7$ is used when reaching the goal state. Clear gains are observed in figure 5a especially at training steps above ~ 250, where the distance to the goal converges in a more stable manner, as well as actually reaching the goal state ($d = 0$).

Question 5 (ii): A sparse reward function would likely fail to improve the basic reward function of $r = 1 - d$ (tried it, diverged). The reason is that we have short episodes (20 steps) so it is hard for the agent to reach the goal often enough to properly back-propagate the rewards to the Q-network. Even if it reaches the goal, ϵ could be too small to fully correct the Q-network and hence the entire Q-value of the grid so the greedy policy would likely deviate from the goal after 500 steps.

1 Figures

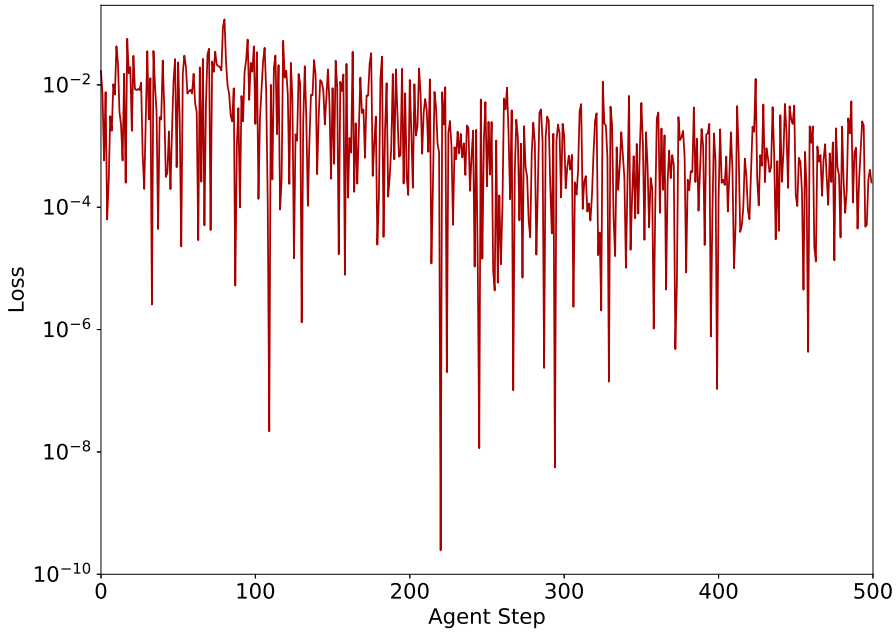

Figure 1a: Agent training step vs. Q-network MSE loss during online learning from a single transition. Instability in training is observed.

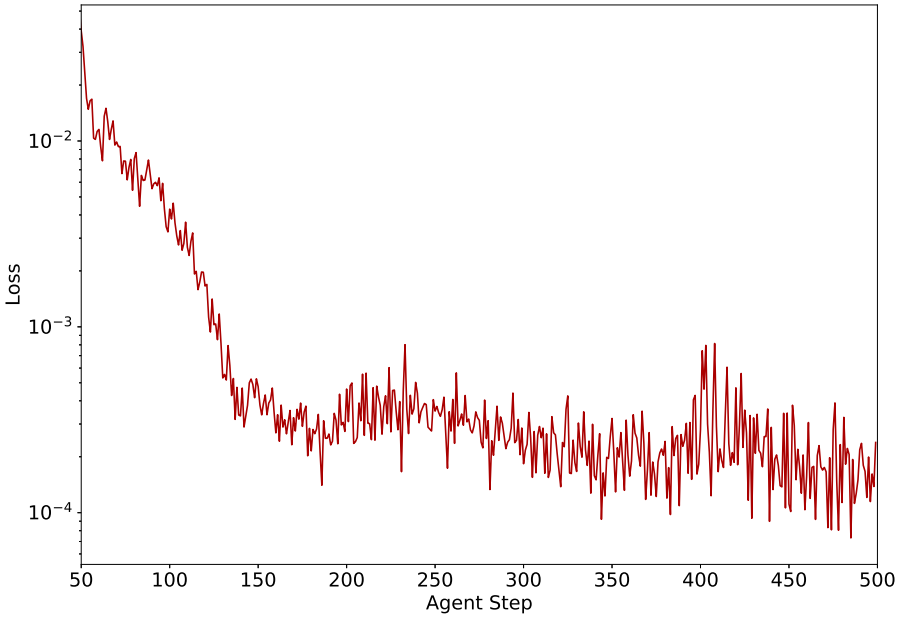

Figure 1b: Training step vs. Q-network MSE loss using an experience replay buffer. The MSE is calculated from the average loss of a mini-batch of 50 steps. Increased stability in the variance is observed as well as convergence behaviour in the MSE loss.

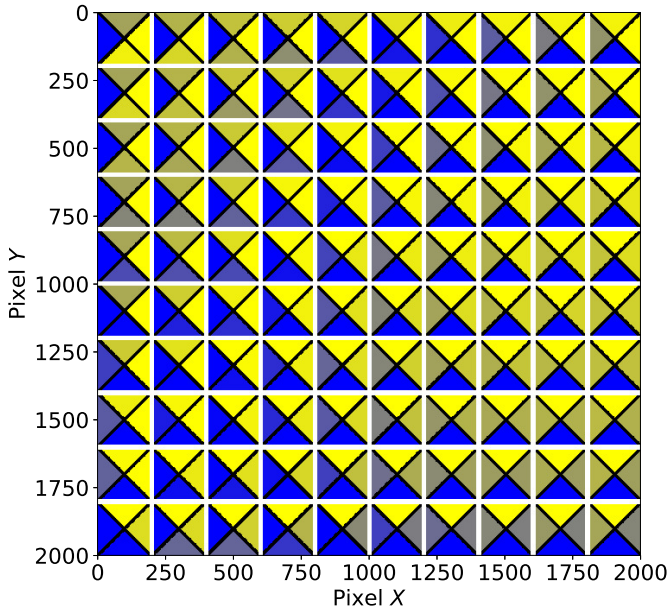

Figure 2a: Q-values using a discount factor of 0 and no target network. Each state is represented by a cell of pixel size (200 by 200) where Q-values for each action (North, East, South, West) are normalised in range $\epsilon\,[0,1]$ and a linear interpolation is performed with yellow representing the most rewarding action and blue representing the least rewarding action. We observe how the Q-values are properly defined nearby the starting position whilst poorly defined further away near the reward state.

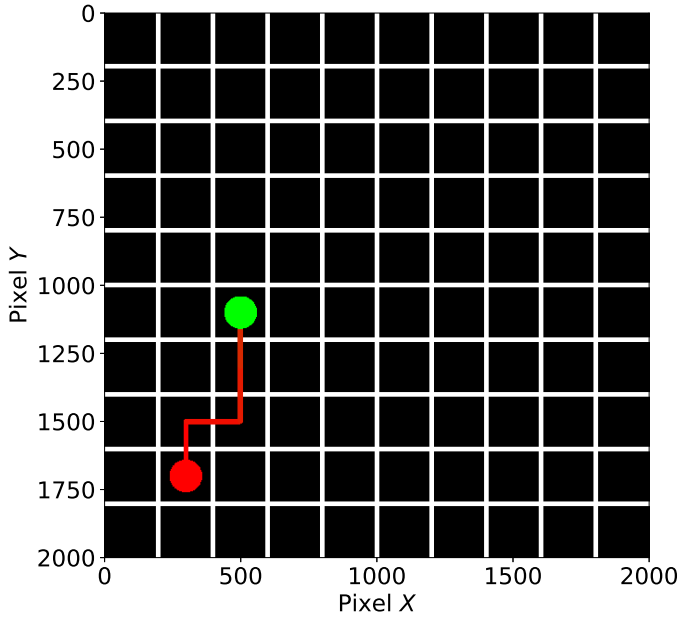

Figure 2b: Greedy policy. The red dot reflects the starting state of the agent whilst the green dot represents the agent's state after 20 greedy steps. The line connecting the dot is interpolated at each state. In this case, the green dot is stuck as it understands the best policy is to move right, however this is interrupted by the "grey block".

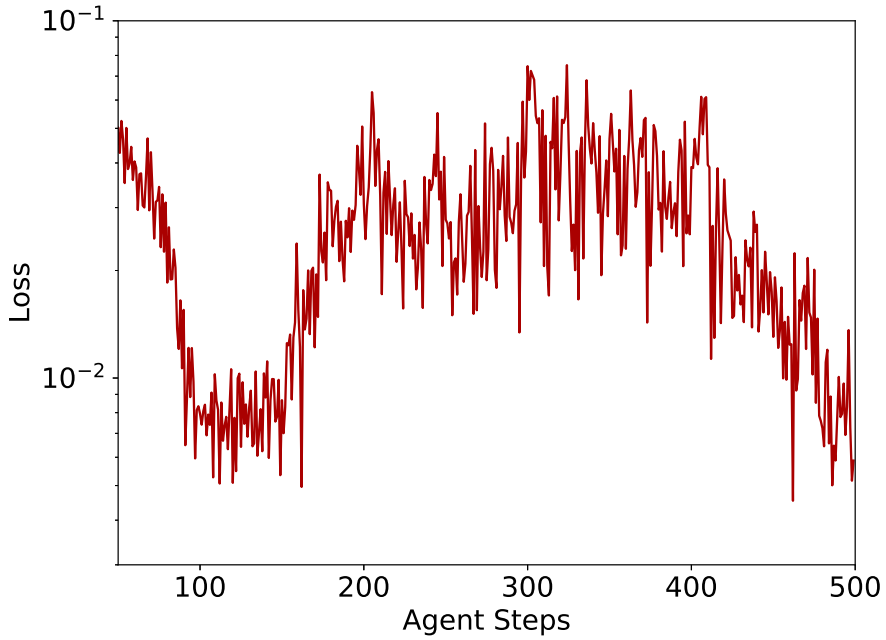

Figure 3a: MSE loss from the Q-network. The loss is unstable and the variance does not seem to reduce. Moreover, convergence appears to behave in a cyclical fashion.

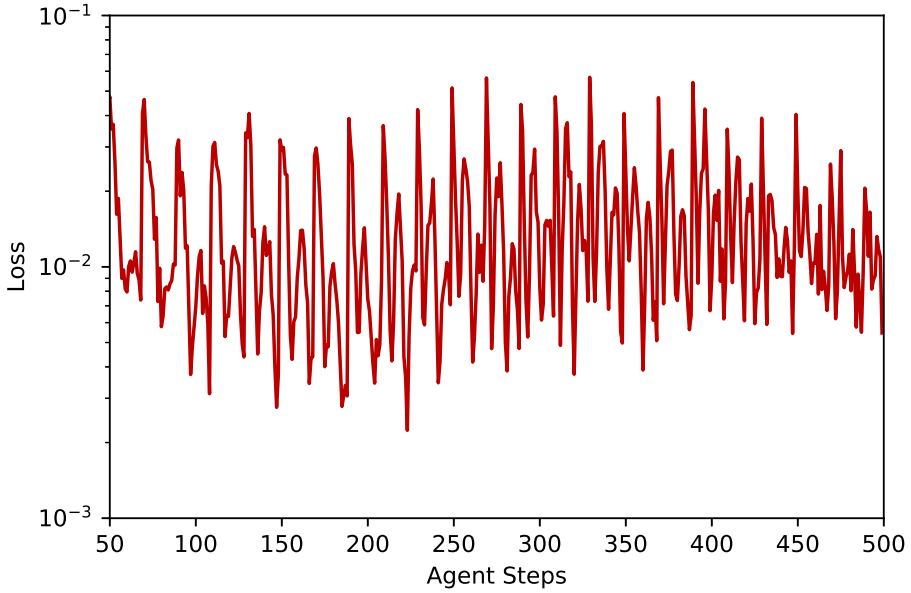

Figure 3b: MSE loss using a target network. Convergence behaviour is contained and the variance of the loss during training slowly but steadily reduces.

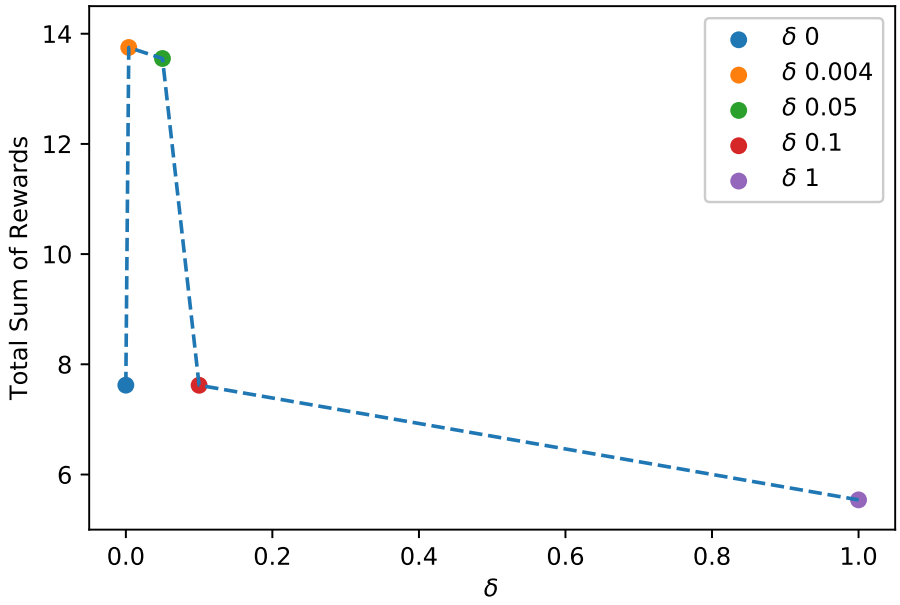

Figure 4: Epsilon decay rate δ vs. total sum of rewards using the greedy policy after 500 training steps. We observe the highest sum of rewards when selecting a δ value of $4e-3$.

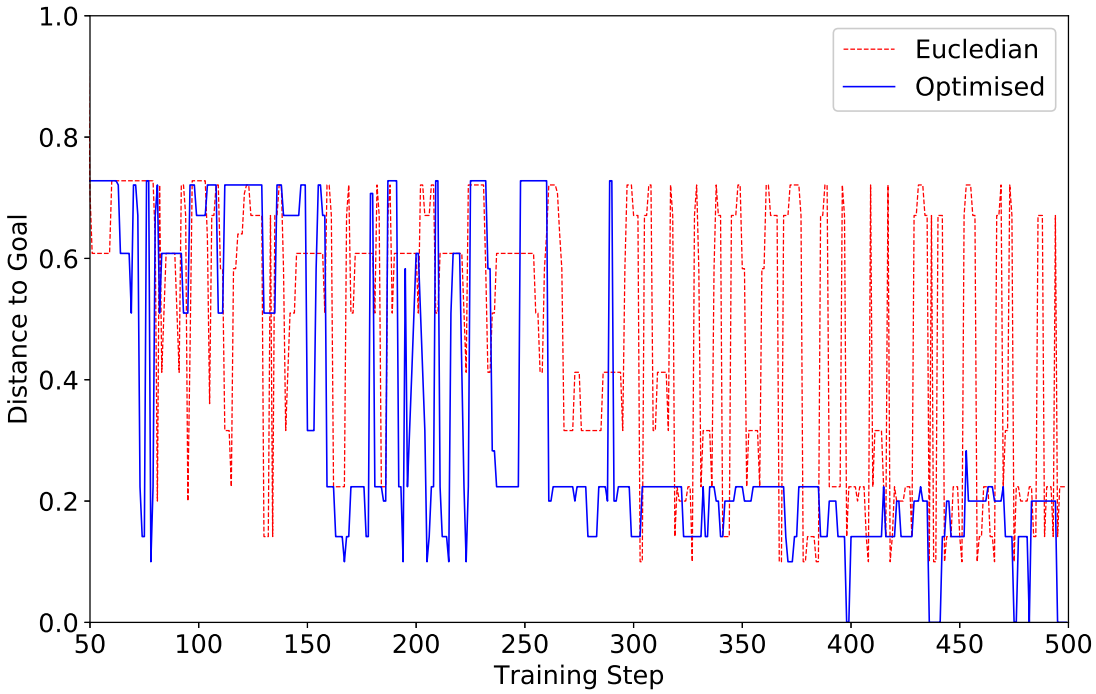

Figure 5a: Training step vs. final distance to goal for the optimised reward function (blue) and the Euclidean reward function $r = 1 - d$ (red).

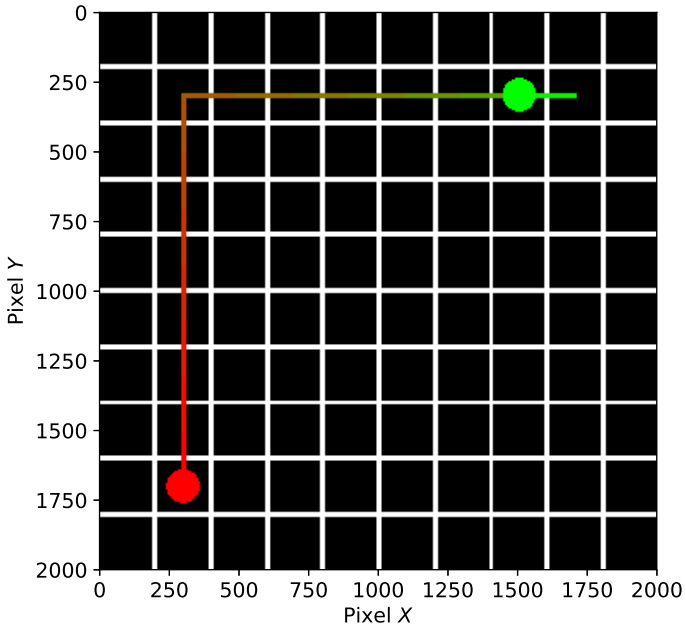

Figure 5b: Visualisation of the greedy policy for the optimal reward function. Here the agent arrives and oscillates around the goal state as it cannot break the loop. This is a clear improvement to Figure 2b, where the agent got stuck to a wall in the previous states.