A.I. In games individual assignment

Student : 180332423

Q1)

Epsilon Greedy:

This is an algorithm which aims at balancing the exploration against the exploitation. It selects a random action/s whose probability is given by epsilon. The actions the agent learns have a larger reward immediately after when trained using a greedy policy, which can lead to ultimately having better reward at successive states. This can however not always be optimal therefore randomizing actions with epsilon-probability gives a broader range of actions to explore and a better policy for the agent to learn, making it necessary to act according to an epsilon greedy Policy.

Q2)

Online Q-learning: here updates to the Q network which increase Q(s,a) have the effect of also increasing Q(s+1,a) for every action, ultimately resulting in an higher values of y. This can lead to oscillations and divergence; having a Q network with updated parameters as a separate target introduces a time delay from the time an update to the Q network takes place and the time this is reflected in the y target.

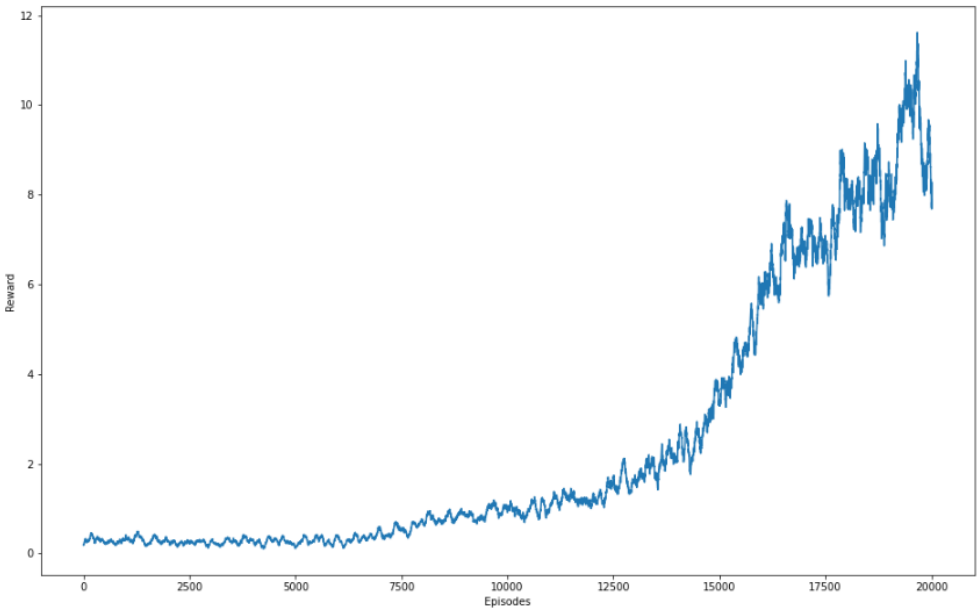
Q3)

When the current frame is the final frame, the baseline implementation of the one-step return, calculates the modified/updated value of q as –1, though, if the current frame is the last frame, the updated value of q should be the reward obtained in the final frame, according to the one step return formula. This result can be –1 or not.

In the baseline implementation, the q value at the final frame is calculated as 1, thus assuming that the final frame reward is -1 too; which creates no apparent issue with the game Atari, but does not work for different Atari games since they might have reward at the final frame different from -1. Therefore, the implementation must be modified and adapted.

By multiplying future rewards with zero and returning only the reward, the right implementation of the one step return calculates the updated q values to be the reward gained at the last frame.

Q4)



Q5)

WarpFrame

The WarpFrame wrapper is used to change the size of a frame to specified values of width and height whilst also converting it to greyscale of required. Its parameters are: width value, height value, and a Boolean specifying whether or not it should be saved as greyscale. The other parameter is dict_space_key which specifies the observation used to be warped if dictionary observations are used by the environment.

ScaledFloatFrame

This wrapper converts the observation between 0 and 1: it achieves this by dividing each pixel value by 255, thus returning a value between 0 and 1 since the images have a pixel intensity ranging from 0 to 255, the value 255 is used as in case of a pixel with intensity 255, a value of 1 is returned.

FrameStack

If the information which is provided by a frame is incomplete, multiple frames might be combined into a FrameStack, which might be able to provide the agent with further information about the current environment. In the example of Atari, it isn't possible for the agent to judge the direction of the motion of the ball by using a single frame, therefore in order to allow the agent to work optimally, FrameStack provides the agent with a stack of the last k frames stored in a dequeue.

MaxAndSkipEnv

The MaxAndSkipEnv wrapper is used to skip several frames and then returns them as well as a reward obtained by the frames which have been skipped. The number of skipped frames is given as a parameter and the skipped frames are returned together with the maximum frame over the latest observations, which are stored in the buffer.

ClipRewardEnv

The ClipRewardEnv wrapper is used to return 1 if the reward is appositive value, while it returns -1 if the reward is negative. If the reward is 0 this returns zero, therefore allowing for some form of regularisation in case the actual value of the reward is not required or in case they need to be encoded by only 3 values: -1, 0 and 1.

EpisodicLifeEnv

This wrapper keeps track of the agent's lives in the game and only resets the game environment when the agent runs out of them. When the agent loses a life, the wrapper marks it as the end of the episode and performs a no-op operation to exit the terminal state rather than resetting the game environment, and operation that deletes the state information.