Ole Christer Selvig - ocselvig@stud.ntnu.no
Eirik Ek Folkestad - oefolkes@stud.ntnu.no

AI Programming - Project 2

Exercise 2a *Implement an agent with this Q function (Demo). Explain what data structure you use to represent the function (Report in writing)*

The Q function for the implementation of Q Learning used in the Frozen Lake environment is a two dimensional array. The first dimension represents all possible states, while the second is the action value for each possible action in a specific state.

Exercise 2c *Keeping the Q function fixed, demonstrate an agent following an ε-greedy policy (Demo). Does this policy present an opportunity for learning a better Q function? If so, why?*

An ε-greedy policy presents a better opportunity for learning a better Q function, because it will provide more exploration by choosing random actions. Adjusting the epsilon factor over time can provide more exploration in the early stages of the learning phase, and more exploitation towards the end.

Exercise 2d: *For the start state s (obtained from resetting the environment), take action South (you may have to first figure out which of the 4 discrete actions corresponds to going South), and follow an ε-greedy policy thereafter (based on our Q function). Accumulate immediate rewards for 100 steps or until the end of an episode (whichever is earlier). Let the discounting be 0.99.*

Exercise 2d: *In statistical terms, what is the relationship between this reward quantity just accumulated and Q(s,South)?*

The reward quantity in it's means is a feedback on whether or not the actions had any positive or negative outcome for the agent. This feedback can be related to probabilistic adjustments of each state-action pair. With a high discount factor, the agent will be drawn towards future rewards, i.e. the accumulated reward will be more significant for determining the value of Q(s, South). It will either raise or lower the probability of following the same trajectory again. After one episode, the value of Q(s, South) will either be a better or worse choice than before, in a sense, depending on the accumulated reward of the trajectory.

Exercise 2d: *Can such quantities help make better estimates of Q(s,South) or for that matter Q(s, a) for any state s and any action a?*

Yes, after sampling many episodes the Q(s,a) estimates on a trajectory will converge towards Its optimal state-action value. The relationship between the estimate and the accumulated quantity is related by how the future rewards are weighted, which determined by the discount factor.

Ole Christer Selvig - ocselvig@stud.ntnu.no
Eirik Ek Folkestad - oefolkes@stud.ntnu.no

Exercise 2d: *What if we follow a greedy policy instead?*

Yes it will provide correct estimates for Q(s, South) and Q(s, a). However, it will focus more on Q(s, South) until it reaches a point where Q(s, South) < Q(s, a). We would recommend optimistic initial values for Q-function for this approach to work, as it provides more exploration.

Exercise 4: *Instead of arg maxa Q(s' , a) as part of the target, where s' is the next state, let us now use the value of the next chosen (by our ε-greedy policy) action instead. Demonstrate the new update rule on FrozenLake (Demo). How is this algorithm different from Q-learning?*

This algorithm, which is SARSA, is different form Q-learning by not assuming the best next action value for learning. Instead SARSA learns from its next action selected by it's current policy. SARSAs learning is dependent on the policy being followed and is therefore on-policy learning.

Exercise 5: *Instead of arg maxa Q(s' , a) as part of the target, where s' is the next state, or the value of the next chosen action, let us now use max action values from the final Q function of exercise 3 as part of the target (Demo). Is this on-policy or off-policy learning?*

This is off-policy learning because the learning function is not dependent on the current policy being used. It assumes that the best action value from what the other algorithm has learned. This will be equivalent to assuming the best action value rather than selecting the action value from a state selected by the current policy.

Exercise 6: *Let's say we did not want an agent to learn from direct interactions with the environment from the word go, but instead train it to some level of feasible behaviour before putting it in the wild. This means we do not have direct access to the environmental dynamics, i.e. the immediate rewards from taking an action in some state, and the state to which the agent transits to upon taking the action. However, we may have interaction data from the past. For example, say a company were to interact with customers to offer a range of products over time. But, the company did not know what the correct sequence of such offers should be for each customer. Each customer is different, some may not be happy getting offers at all, while others like being offered products in accordance with their needs. Offering products they do not need would annoy them. Not offering products they may need, which they then come to know about themselves, may disappoint them. One way to find a policy for the right sequence of products per user would be to let an agent simply learn by interacting with the environment, i.e. customers, but this could be pretty expensive for the company as bad offers may lead to customers getting annoyed and ceasing to use the products the company offers. However, let's assume that the company has interacted with its customers in the past through human operators (e.g. in call centres), who listen to the customer and figure out their situation, record this situation, offer a relevant product that tries to better their situation, and record feedback on how this offer was perceived by the customer. Can we see the human agent as a policy that generates data an artificial agent can learn from? If so, would you use an on policy or off policy learning algorithm to train the agent on this data?*

Yes, we can see the human agent as a policy that generates data an artificial agent can learn from. However, this does not mean that the data offers an optimal policy, i.e. that the data might not contain definite information on what is the best course of actions. On the other hand we believe that the the learning agent should do the least amount of errors while training. Off-policy

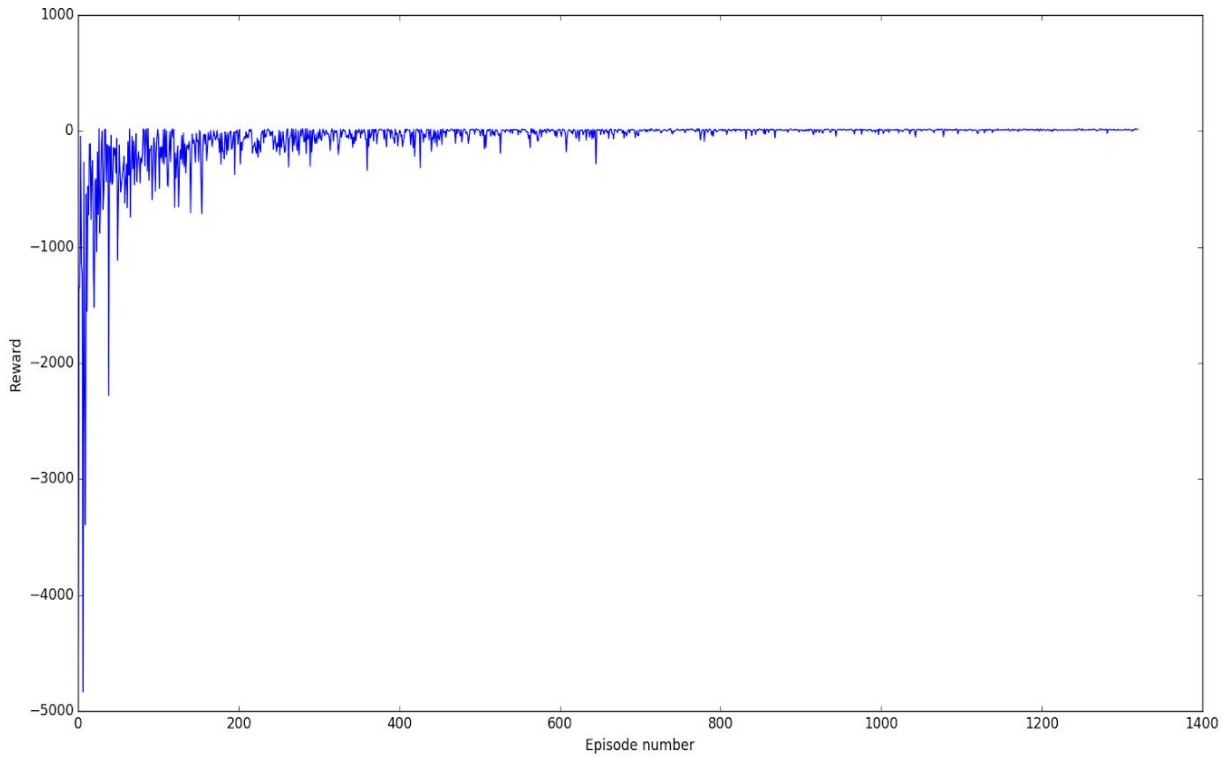
Ole Christer Selvig - ocselvig@stud.ntnu.no
Eirik Ek Folkestad - oefolkes@stud.ntnu.no

learning may be faster to find the optimal sequence for each customer but on-policy would be more careful in the process and should annoy the customers less. We would, from this reason, choose on-policy learning over off-policy.

Exercise 7: *Apply Q-learning and the method implemented in exercise 4 on Taxi. Make use of an ε-greedy policy where ε = 0.1, a discounting of 0.99. You are free to tweak the learning rate (keeping it within [0.0, 1.0)) and the number of episodes you run the two learning algorithms for (Demo). Explain what data structure you use to represent the Q function.*

For this problem we used python dictionary for a more efficient representation for the Q function, as the update for a state-action value is constant O(1). The state-action space is bigger for this particular problem, therefore we reasoned that this would give us a performance advantage.

Exercise 7: *Show the performance of your algorithms by plotting the total reward per episode against the episode number.*



Ole Christer Selvig - ocselvig@stud.ntnu.no
Eirik Ek Folkestad - oefolkes@stud.ntnu.no

Figure information:

Last 100 elements of reward_list: [15, 13, 15, 19, 10, 10, 12, -2, 8, 7, 8, 12, 14, 9, 11, 10, 15, 11, 7, 7, 8, 10, 6, 8, 10, 14, 6, 6, 7, 13, 9, 13, 12, 10, 13, 13, 14, 10, -21, 16, 14, 8, 8, 14, 14, 10, 16, 18, 7, 6, 8, 1, 7, 12, 12, 18, 11, 12, 13, 10, 10, 14, 8, 7, 9, 8, 9, 5, 6, 19, 7, 9, 12, 18, 17, 15, 3, 8, 14, 6, 9, 7, 9, 7, 6, 7, 10, 8, 12, 6, 8, 8, 4, 7, 12, 9, 9, 7, 9, 9]

Reward_list length: 1321

Average reward of last 100 episodes: 9.74

Episode 1320