NTNU – Trondheim
Norwegian University of
Science and Technology

IT3105 - AI Programming

Project 2 Report

Reinforcement Learning

Annie Aasen
Mikael Bjerga

October 18, 2016

Exercise 2

Exercise 2a

To represent the Q-function, we use a two-dimensional numpy array. The rows correspond to the different actions possible, while the columns correspond to the observations/states. This means that an entry in the array corresponds to a specific action taken in a certain state.

Exercise 2c

Yes, the epsilon-greedy policy present an opportunity for learning a better Q-function. The policy allows for some degree of exploration because the algorithm will choose a random action with the probability of epsilon. The pure greedy policy will always choose the action with the highest Q-value and will thus never be able to explore the outcome of taking the other actions. The best way to learn a better Q-function is to both explore and exploit.

Exercise 2d

The value of Q(s,South) represents the expected reward of taking the action South in an arbitrary state s. The reward quantity just accumulated is the actual reward received in the end by taking the action South when in the initial state. This means that in statistical terms, Q(s,South) is the expected value while the reward quantity accumulated is the actual value.

Using such quantities to make better estimates of Q(s,South) or for that matter Q(s,a) for any state s and action a could be problematic following an epsilon-greedy policy. The later rewards in a long chain of rewards are more likely to be uncertain than the earlier ones. Usually, when using rewards from several steps to update a Q-value, the rewards are discounted proportionally to how many steps are needed to gain them. In this case, however, the accumulated sum is discounted as a single future step. The uncertainty of the rewards may then be transferred to the estimate. It is especially uncertain when using random actions to explore, as in the epsilon-greedy policy. The rewards accumulated when exploring may not be representative for the agent, making the not properly discounted rewards even more untrustworthy.

Following a greedy policy, the algorithm will always choose the assumed optimal action for every run. While the later rewards are still more uncertain than the earlier ones, the accumulated rewards would be more helpful when using a greedy policy instead of epsilon-greedy. This is because we do not use random actions to explore, which would add uncertainty. Thus, we know that the rewards given are a result of actions that will actually be chosen after the agent has moved south (disregarding slipping).

In both the greedy and the epsilon-greedy case, updating with the accumulated rewards without properly discounting them will give each sample run too much influence on the Q-value of the South-action in the initial state, as the run may not be representative. Using several sample runs with proper discounts will create a better estimate, as the average run rewards will balance out the abnormal run rewards. Using proper discounting will also encourage a shorter path to the goal.

Exercise 4

The algorithm in this exercise replaces the value of the assumed next action taken with the value of the actual next action taken. Depending on the value of epsilon, these actions will sometimes be the same, but not in the case of exploration. If the algorithm chooses to explore (with probability epsilon), Q-learning will still update the Q-value based on the assumed best next action and not the actual next action taken. This is called off-policy learning. The new algorithm described will always update the Q-value based on the actual action chosen next, whether it is the best one or not. This is called on-policy learning.

Exercise 6

Yes, the human agent can be seen as a policy that generates data an artificial agent can learn from, as the operator is in a state (with a particular customer that has a previous offer sequence), tries an action (offers a product), and records the reward (feedback). This can then be used by the agent to learn.

Assuming the right sequence of products for a customer does not change much over a short period of time, we would use an on-policy learning algorithm to train the agent. The assumption of following an optimal policy, as done in off-policy learning, could be said to be unreasonable, as we are dealing with highly complex agents in the human customers. Thus it is unlikely that the operators are following an optimal policy, and therefore on-policy learning would provide a more reasonable approach, as there is more focus on the actual feedback from customers on different offer sequences.

However, due to the complexity of the human customers, one could argue that off-policy learning would be better suited to deal with the change-prone and seemingly randomness of human behaviour, where one offer could be well received one day, but rejected another, due to circumstances beyond our detection and control, such as mood and priority changes.

Exercise 7

We have chosen the same representation of the Q-function for the Taxi environment as in the Frozen lake environment. We use a two-dimensional numpy array with rows corresponding to the different actions possible, and columns corresponding to the observations/states. This means that an entry in the array corresponds to a specific action taken in a certain state.

On the next page there are two figures showing the total reward per episode plotted against the episode number. You can see that in this environment, the Q-learning off-policy algorithm performed slightly better than the SARSA on-policy algorithm.

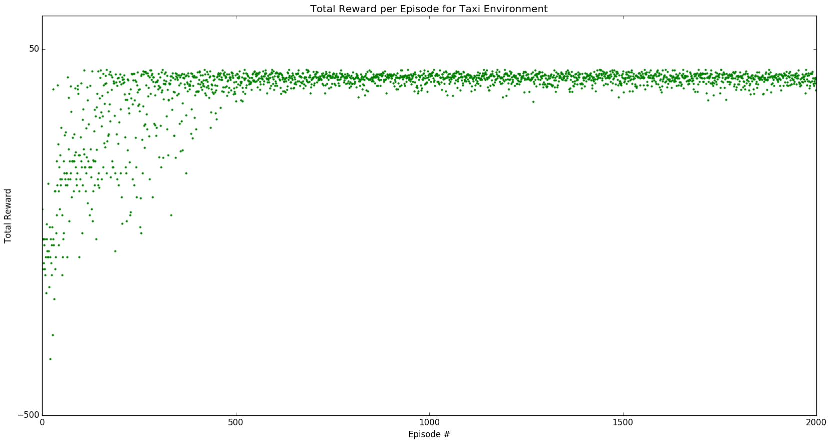

Figure 0.1: Off-policy (Q-learning)

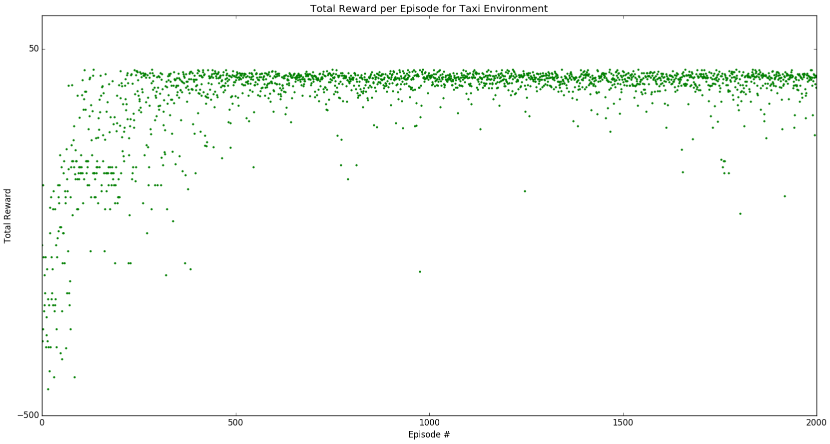

Figure 0.2: On-policy (SARSA)