Reinforcement Learning Lab Session 6

Your Name Here

April 2, 2020

General Information

This lab session will cover all the Reinforcement Learning material from courses 1 to 4. It will be graded lab sessions. You are free to use any content you want regarding the questions, but please refrain from using outside code beyond the previous lab sessions.

It should be doable in 4 hours, but the deadline will be set to Thursday 2nd of April, midnight. As usual, for there on, each day of delay will remove 2.5/10 points from your grade.

Submission – You will have to submit both a report and code through Blackboard. Use the code available on the git at `http://github.com/Louis-Bagot/RL_Lab/lab_session6` or `https://github.com/Rajawat23/RL_Lab/lab_session6`.

Make a copy of this LaTeX document from folder **report**, and fill it according to the instructions below. Please to not alter this base document, only add your content.

Question – *Questions will look like this. For questions, write the answer just below where the (your answer here) appears.*

Programming – *Programming exercises will look like this. For programming exercises, read the `instructions.md` of the corresponding section, and then fill in the corresponding TODO flags in the code. If this text asks for a plot, copy the plot output of your code in the document, in the figure space provided (in addition to leaving it in the plots folder in the code). If the text asks for an explanation of the results, write it as you would answer a question.*

Contents

1 Bandits

Question 1 – *Explain, in a few lines, the k-armed Bandit problem. Give an example of a real-world application, detailing actions and rewards.*

In the k-armed Bandit problem, you are choosing between a number of one-armed bandits, each with different probabilities of rewards, and you want to choose the best one.

Question 2 – *Derive the incremental updates of the Sample Average method.*

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$Q_{n+1} = \frac{1}{n}[R_n + \sum_{i=1}^{n-1} R_i]$$

$$Q_{n+1} = \frac{1}{n}[R_n + \frac{n-1}{n-1}\sum_{i=1}^{n-1} R_i]$$

$$Q_{n+1} = \frac{1}{n}[R_n + (n-1)Q_n]$$

$$Q_{n+1} = \frac{1}{n}[R_n + nQ_n - Q_n]$$

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

Question 3 – *Explain, with your own words, the difference between Sample Average and Weighted Average methods*

In the Sample Average method each reward will carry the same weight, whereas in the Weighted Average method more weight is given to recent rewards. While the Weighted Average method does not converge completely, it gets there faster than the Sample Average.

Question 4 – *Explain the impact of the hyper-parameters of the following algorithms: ϵ in ϵ-greedy, c in UCB, and α in Gradient Bandit. Use extreme values as examples.*

The ϵ in ϵ-greedy determines with which probability ϵ we will take a random action. A very high ϵ results in high exploration, meaning we will get performance nearing the Random algorithm. A low ϵ results in high exploration, giving us a low chance to explore possibly better actions.

The c in UCB determines how much of the potential of each action we will take into account. A very high c will result in almost always taking the least used action, which gives equal performance to the Random algorithm. For a low c we will not take the actions potential into account, only it's estimated reward. This is equivalent to the greedy approach.

The α in Gradient Bandit determines how much the preference for all actions changes given a reward of a certain action. A very high α results in the preference changing drastically. Giving a high preference for actions with a high reward. Similair to a more greedy approach. While with a very low α the change in preference is extremely low, resulting in a more Random approach.

Question 5 – *Show that the Sample Average method erases the bias introduced by the initialization of the Q estimates. What does this mean for Optimistic Greedy? Show that, despite this, Optimistic Greedy does not work on non-stationary problems.*

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

$$Q_2 = Q_1 + \frac{1}{1}[R_2 - Q_1]$$

$$Q_2 = Q_1 + R_2 - Q_1$$

$$Q_2 = R_2$$

We now see that the bias introduced in Q_1 is erased from Q_2.

For Optimistic Greedy this means that we will first take each of the actions once, given a high enough initial Q setting. Afterwards the algorithm would continue as if it was a normal Epsilon Greedy algorithm.

Programming – *Implement a Sample Average and Weighted Average version of ϵ greedy on a Non-Stationary k-armed Bandit problem. In order to see results, run the experiment for $10k$ steps, and paste here the resulting performance plot in the Figure 1 below. Explain your results below.*

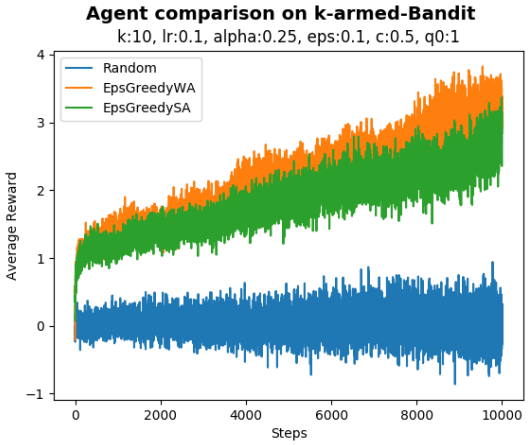

Figure 1: Comparison: ϵ-greedy algorithm with Sample Average versus Weighted Average updates.

2 Markov Decision Processes

For questions where a drawing (MDP or diagram) is required, you can use whichever of the following methods:

- a (properly cropped and clear) photo of a drawing on paper. Make sure everything is readable.
- a tikz diagram, i.e. the plotting tool for LaTeX (if you know how it works. Don't learn for this report otherwise, tikz takes an eternity)
- Mathcha, which can generate tikz or pngs. (recommended)

Question 1 – *Define a Markov Decision Process, and the goal of the Reinforcement Learning problem.*
A Markov Decision Processs is a 4-tuple (S, A, R_a, P_a), where

- S is a finite set of states
- A is a finite set of actions
- R_a is a reward we get because of taking an action in a state
- P_a is a probability function $p(s', r|s, a)$

The goal of the Reinforcement Learning problem is to find a policy which describes the best action for each state in the MDP, known as the optimal policy.

Question 2 – *Show that the MDP framework generalizes over Bandits by drawing the Bandits problem as a MDP with reward distributions r_a for each action a. Paste your drawing on Figure 2. Shortly explain your submission.*

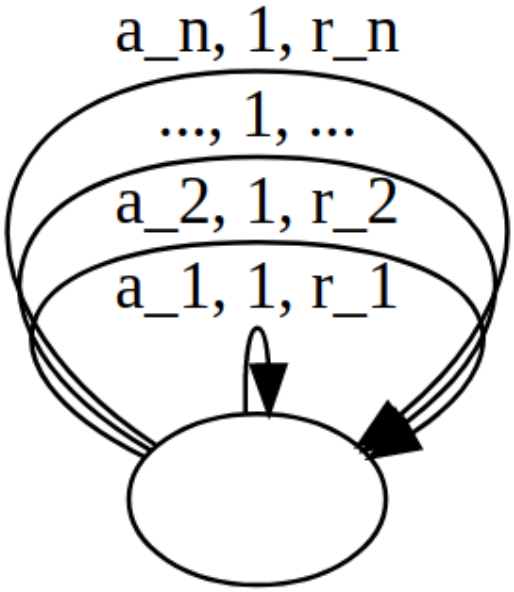

Figure 2: The MDP corresponding to the Bandit problem with reward distributions r_a, $\forall$ actions a

.

In bandit problems, the reward for a given action does not depend on which arms were pulled previously. In an MDP this can be modelled by using only 1 state from which all the different actions can be taken.

Question 3 – *Turn the following statement into a MDP, with states and transitions with actions (named), probabilities and rewards. Paste the graph on Figure 3; pick real values for both rewards and probabilities (no unknowns). Shortly explain your submission after the statement and plot.*

Statement:

You go to the university using Velo - Antwerp's shared bikes. There are three stations where you can drop off your bike on the way: the park, furthest from the university; the cemetery, second furthest; and the university station, right in front of your destination. You want to take the least time possible to go to the university, and you take much longer walking than biking.
At any station, you can either decide to drop off your bike and walk to the university, or continue to the next station. However, it sometimes happens that the stations are full - you cannot drop off your bike there. You can either go back, or, if possible, continue. You notice that the amount of free spots in the first stations often aligns with the amount of free spots in the following stations - or is less. In order to decide whether you should drop off your bike or not, you take note of the last station's number of free spots - it can either be a lot, or a few, or none.
When you have to go back, we assume that people could've come to pick or drop bikes, so the transition doesn't depend on the previous state of the station.

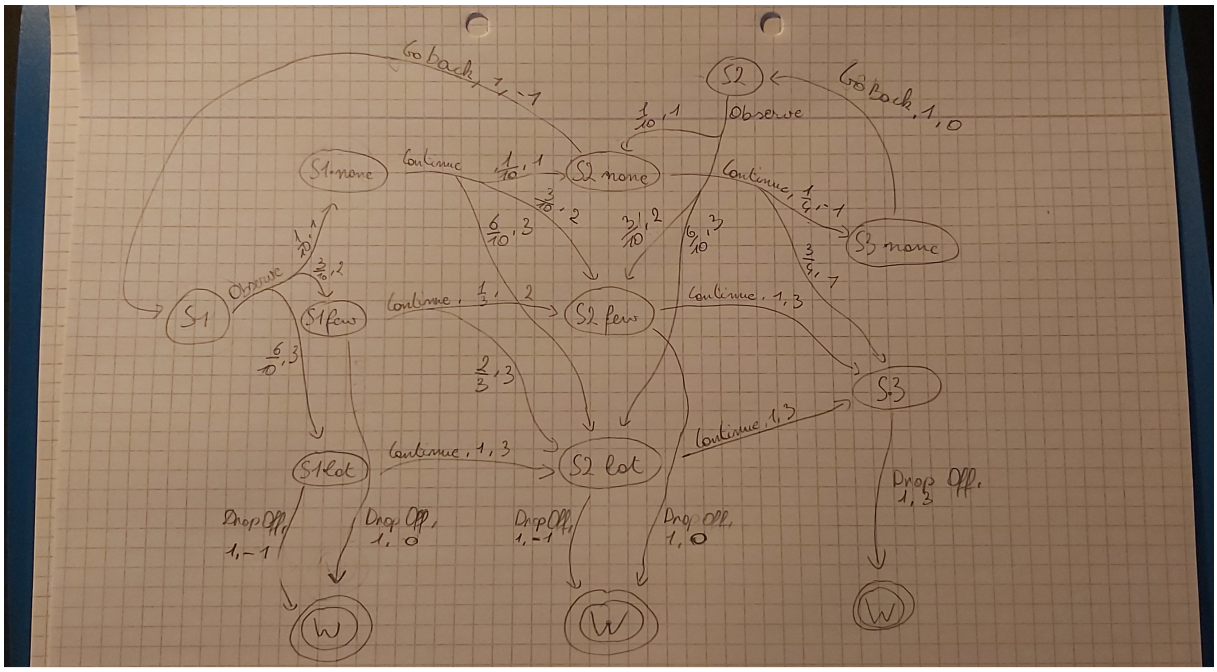

Figure 3: The MDP corresponding to the statement of Question 3

.

The first 2 stations have nodes, "S none", "S few" and "S lot" to represent the amount of free spots. The third station only needs nodes for when there are no empty spots and when there are. From these nodes, you can either continue to the next station (next set of nodes) or drop off your bike and walk to the university by going to the "W" node. When you decide to go back to the previous station, you arrive at either node "S1" or "S2".

Question 4 – *RL has been widely popular lately because of its combination with Deep Learning (using Neural Nets as policy or value function approximators), leading to incredible performances on difficult environments like video games. One such game is the first Mario World. Show how the MDP can look like for the frames of the game. Where can stochasticity be involved?*

3 Control

In lab session 2 and 3, the Value Iteration algorithm was stopped after a maximum number of iterations. However, this is not the natural stopping condition of the algorithm: it should stop when the value estimates have converged: $V_k = v*$. When implementing this, we define convergence of the V_{k-1}, V_k, V_{k+1}.. stream of vector values as

$$||V_{k+1} - V_k||_2 < \delta$$

Where δ is an arbitrary small constant ($\delta = 0.01$ in the code). The number of iterations of Value Iteration to convergence is a measure of the algorithm's performance.

Policy Iteration alternates evaluating a policy π until convergence to V_π, and updating the policy to be greedy over the new values, $\pi' = greedy(v_\pi)$. We define *convergence in policy* as $\pi' = \pi$ (same action in all states), and *convergence in value* as

$$||V_{\pi'} - V_\pi||_2 < \delta$$

Value Iteration only converges in value, as there is no explicit policy. When comparing convergence speed in value of Value Iteration vs Policy Iteration, make sure to compare the number of single sweeps over the state space! (iterations)

Programming – *Implement Value Iteration on the course's diamond/pit Gridworld environment (course and Lab session 2). You can reuse Environment code from before.*

Programming – *Implement Policy Iteration on the course's diamond/pit Gridworld environment.*

Question 1 – *Discuss and compare the performances of both algorithms. Under what circumstances can one come on top of the other?*

For Value Iteration I observed 11 sweeps over the state space, while for Policy Iteration there where 29 sweeps. In Policy Iteration, the policy converged after 3 iterations of Policy Evaluation and Imporvement. This means that in most of the sweeps in Policy Iteration, we did not have to iterate over all the possible actions for each state.

Question 3 – *Explain the fundamental differences between QLearning and the Value Iteration / Policy Iteration algorithms. Can you see why QLearning is more interesting in the general case?*

QLearning is model-free where training samples the transitions (s, a, s', r), this means that the agent does not know the state transition probabilities or rewards. Only by doing an action from one state to another can the agent discover the reward for doing so.

4 Bonus

Programming – ***BONUS, 1.5pts*** *Implement the Gridworld drawn on Figure 4: a river crossing. The actions are up, down, left, right and "do nothing". The agent needs to cross a river from the lower left corner (state S) to the upper right corner (state G). This* 3×5 *Gridworld is divided on the 2nd row by a river that might push the agent right upon entering by one, two or three squares, with corresponding probabilities* $0.2, 0.5$ *and* 0.3*. If the agent is pushed off to the far right in the river, the episode ends with reward* -1*. If the agent reaches the goal, the reward is* $+1$*. Note that it is the transition to the state (i.e. "right" from (0,3) to G=(0,4)) that yields the reward, and states G and red (1,4) are terminal.*

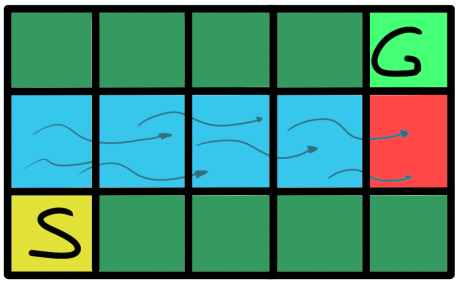

Figure 4: The River Crossing MDP

.