Multi-Agent Systems

Chi Him Ng (2748786)

HW6: Final Homework Assignment MSc Al, VU

Version: December 3, 2023- Deadline: Friday, 22 December 2023 (23h59)

IMPORTANT

- This project is an individual assignment. So everyone should hand in their own copy.
- The questions below require programming. In addition to the report (addressing the questions and discussing the results of your experiments), also provide the code on which your results are based (preferably as Python notebooks). However, make sure that the report is self-contained, i.e. that it can be read and understood without the need to refer to the code. Only the report will be graded, so it's NOT sufficient to submit only a notebook.
- Store the pdf-report and the code in a zipped folder that you upload to canvas. Use your name and VU ID as name for the zipped folder, e.g.
- This assignment will be graded. The max score is 4 and will count towards your final grade.
- Your final grade (on 10) will be computed as follows:

Assignments 1 thru 5(max 1)+ Individual assignment (max 4) + Final exam (max 5)

- Good luck!

1 Monte Carlo Simulation (30%)

The COVID scare was a wake-up call, highlighting the vulnerability of our hyper-connected modern world to the threat of pandemics. Now that the COVID urgency has passed, governments across the globe are rushing to update their emergency action plans and disaster scripts.

As part of this effort, you've been hired by a medical team that has been tasked with developing fast procedures to detect a blood-borne virus. Since these tests need to be administered to large groups in the population, and testing resources are limited, the medics have come up with the following procedure. They started from the assumption that they need to test N blood samples (of as many different individuals) and that N is large (e.g. $N = 10^6$). Furthermore, the probability that an individual is infected is p, where p is relatively small, e.g. $p < 0.1$.

Based on these assumptions they propose the following procedure to minimise the number of tests they have to run: Rather than testing each sample individually, take a batch of k samples and mix them. Then this mixed sample is tested for the presence of the viral antigen:

- If the mixed sample tests negative (i.e. no viral antigen is detected), then all the individual samples were clear, and you therefore have the result for all k individual samples that went into the batch.
- If the mixed sample tests positive (i.e. the viral antigen is present indicating infection), one needs to retest all individual samples that went into the batch, in order to find out which individual(s) are actually infected.

Questions

1. Use Monte Carlo simulation to estimate the optimal batch size k (i.e. the one that minimises the expected number of tests) for a given value of p where p can take values between 10^{-1} and 10^{-4}.

   ### Answers

   The table displays the p-values alongside the corresponding optimal batch sizes and their expected number of tests required. The optimal batch size (k) represents the number of individual blood samples grouped together and tested collectively in a batch. As can be seen batch-sizes are often lower then 100. With the exception for p = 0.0001. The implementation was done in colab, in order to use the GPU. Since it took a very long time to simulate. The code systematically explores various batch sizes to minimize the number of tests required. Beginning at size 1, then the largest, then half. It converges on the optimal size.

2. In order to convince your superiors that this a good investment, quantify the expected reduction in workload (compared to testing all samples individually).

   ### Answers

   The reduction in workload achieved through optimized batch testing strategies compared to testing each sample individually is substantial. When

Table 1: Table of Optimal Batch Sizes and Expected Tests

p	Optimal Batch Size	Expected Tests
0.1	31	99 997
0.01	62	10 000
0.001	15	1000
0.0001	500	101
0.00001	1	11

considering various probabilities (p) of infection, the expected reduction in the number of tests needed showcases the efficiency gains:

- For higher probabilities of infection ($p = 0.1, 0.01, 0.001$), the reduction in workload is significant, ranging from approximately 933,333 to 983,870 tests, demonstrating the effectiveness of batch testing in scenarios where infection rates are relatively higher.
- For a lower probability of infection ($p = 0.0001$), the reduction in workload is immense, with nearly 998,000 fewer tests needed compared to individual testing.
- At an extremely low probability of infection ($p = 1 \times 10^{-5}$), which indicates a very low likelihood of an individual being infected, the batch testing strategy doesn't yield a reduction in workload, as the expected number of tests using batch testing matches that of individual testing.

Overall, across various probabilities, the total reduction in workload, amounting to approximately 3,882,944 fewer tests compared to testing each sample individually, underscores the efficiency and resource-saving potential of implementing optimized batch testing methods. This reduction in workload signifies a substantial cost and resource-saving advantage, making a strong case for adopting batch testing strategies.

2 Thompson Sampling for Multi-Armed Bandits (30%)

2.1 Beta distributions to model uncertainty about probabilities

Consider a bandit that for each pull of an arm, produces a binary reward: $r = 1$ (with probability p) or $r = 0$ (with probability $1 - p$). Assuming the bandit has K arms, this means that there are K unknown probabilities $p_1, p_2, \ldots, p_K$ and we need to identify the arm that has the highest probability

$$p^* = \max\{p_1, p_2, \ldots, p_K\}$$

as pulling this arm will result in the highest cumulative reward.

Using the beta-distribution to model the uncertainty on a probability Initially we have no information (total uncertainty) about the values of the probabilities, but each pull of an arm yields a binary outcome (reward), providing some information about the underlying probability, and thus reducing the corresponding uncertainty.

We model our uncertainty about the actual (but unknown) value p using a beta-distribution (cf. https://en.wikipedia.org/wiki/Beta_distribution). This is a (unimodal) probability distribution on the interval $[0,1]$ which depends on two parameters: $\alpha, \beta \geq 1$. The explicit distribution is given by (for α, β integers!):

$$B(x;\alpha,\beta) = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!} x^{\alpha-1}(1-x)^{\beta-1} \quad (\text{ for } 0 \leq x \leq 1)$$

The parameters α and β determine the shape of the distribution:

- If $\alpha = \beta = 1$ then we have the uniform distribution;
- If $\alpha = \beta$ the distribution is symmetric about $x = 1/2$.
- If $\alpha > \beta$ the density is right-leaning (i.e. concentrated in the neighbourhood of 1). In fact, one can compute the mean explicitly:

$$X \sim B(x;\alpha,\beta) \quad \Longrightarrow \quad EX = \frac{\alpha}{\alpha+\beta} = \frac{1}{1+(\beta/\alpha)}$$

indicating that the ratio β/α determines the position of the mean.

- Larger values of α and β produce a more peaked distribution. This follows from the formula for the variance:

$$X \sim B(x;\alpha,\beta) \quad \Longrightarrow \quad \mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

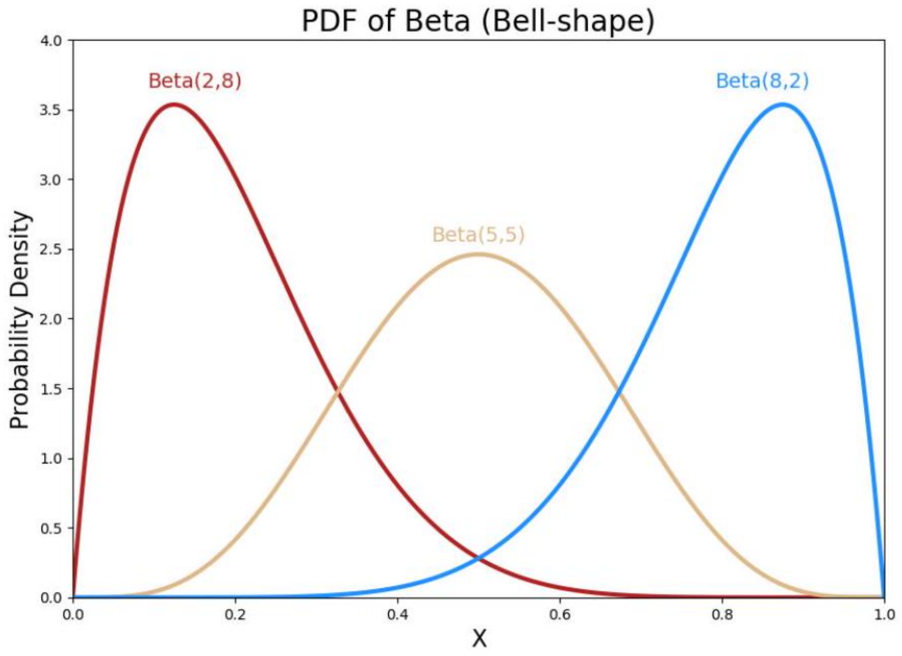


Figure 1: Some probability densities for the Beta-distribution with different parameters.

2.2 Thompson's Bayesian update rule

Although the beta-distribution seems like a reasonable model to quantify the uncertainty on a probability, there is a deeper reason for its use. Updating a (prior) beta-density with binary observations, results in a new beta-density with updated parameters (this is an example of what is known as conjugated priors). Specifically, if the prior is modeled as $B(x, \alpha, \beta)$, and we observe s successes (1) and f failures (0) then the posterior would be the beta distribution $B(x; \alpha+s, \beta+f)$. This observation yields the rationale for Thompson's Bayesian update rule:

- Initialise $\alpha = \beta = 1$ (resulting in a uniform distribution, indicating that all possible values for p are equally likely). Now repeat the following loop:

1. Sample from the bandit and get reward r (either $r = 1$ or $r = 0$);
2. Update the values for α and β as follows:

- if $r = 1$, then $\alpha \leftarrow \alpha + 1$
- if $r = 0$, then $\beta \leftarrow \beta + 1$

This update rule can be summarized as:

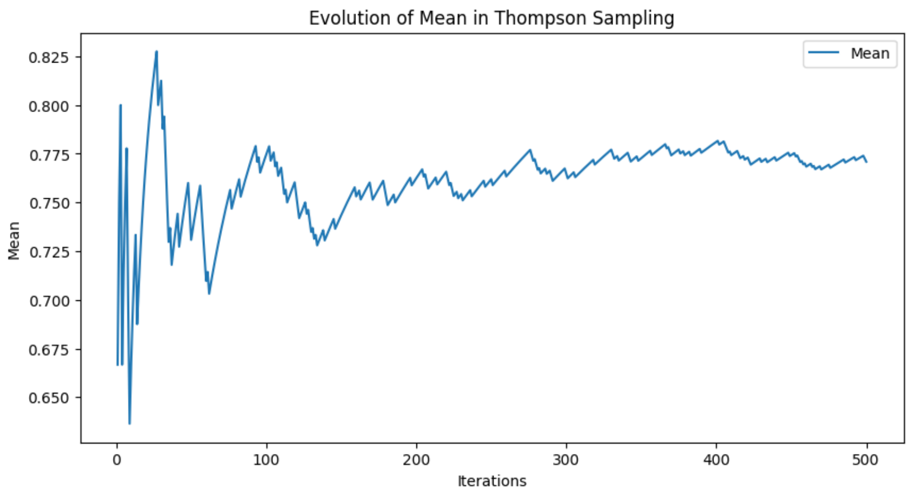
$$\alpha \leftarrow \alpha + r \quad \beta \leftarrow \beta + (1 - r)$$

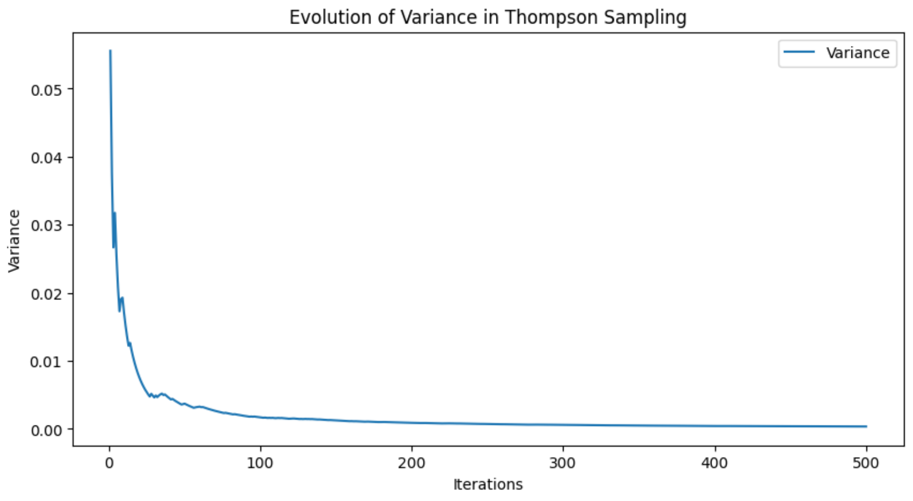
Questions

1. Implement the Thompson update rule for single arm bandit (i.e. $k = 1$) and show experimentally that the Beta-density increasingly peaks at the correct value for p. To this end, plot both the evolution of the mean and variance over (iteration)time.

Answer

As can be seen from the plots, the beta-density indeed peaks at the correct value for p. The variance gets smaller over the iterations.





2.3 Thompson sampling for K-armed bandit Problem

For binary outcomes, the Thompson update rule offers an alternative for the UCB-based balancing of exploration and exploitation. Specifically, suppose we have a K-armed bandit problem. The k-th arm delivers a reward $r = 1$ with (unknown!) probability p_k (and hence $r = 0$ with probability $1 - p_k$). For each arm ($k = 1, \ldots, K$), the uncertainty about the corresponding p_k is modelled using a Beta-distribution $B\left(x; \alpha_k, \beta_k\right)$. Thompson sampling now tries to identify the arem that will deliver the maximal cumulative reward (highest p_k) by proceeding as follows:

Initialise all parameters to $1 : \alpha_k = 1 = \beta_k$; Now repeat the following loop:

- We use the beta-distributions to simulate the pulling of each arm. This means that we sample a value U_k from each of the K Beta-distributions:

$$U_k \sim B\left(x; \alpha_k, \beta_k\right) \quad (k = 1 \ldots K)$$

- For this simulation, determine which arm gave the best result:

$$k_{\max} = \arg\max \left\{U_1, U_2, \ldots, U_K\right\}$$

- Mindful of the uncertainties on the p_k-values, the above simulation gives us reason to believe that pulling the $k_{\max}$ arm is rational (after all, we did a simulation using the available evidence, and this was the result).

- Sample the corresponding arm (i.e. arm $k_{\max}$) and get reward r (either 1 or 0);

- Use the Bayesian update rule for the corresponding parameters:

$$\alpha_{k_{\max}} \leftarrow \alpha_{k_{\max}} + r \quad \text{and} \quad \beta_{k_{\max}} \leftarrow \beta_{k_{\max}} + (1 - r)$$

Questions

2. Write code to implement Thompson sampling for the above scenario when $K = 3$;

Answers

See the attached notebook for implementation, the result was:

Arm	Total Reward
1	404
2	425
3	23

3. Perform numerical experiments in which you compare Thompson sampling with the UCB. Use total regret (provide the precise definition that you're using) as your performance criteria. For UCB, experiment with different values of the hyperparameter c. The fact that, for Thompson sampling, you don't need to specify an hyperparameter, is a distinct advantage.

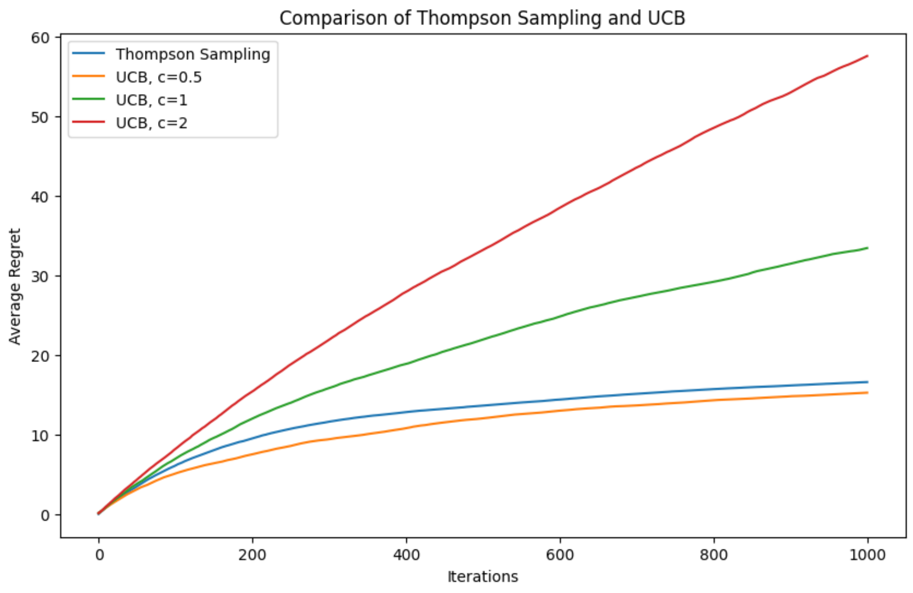
Answers

The performance criterion used in comparing Thompson Sampling and Upper Confidence Bound (UCB) algorithms was the total regret, defined by:

$R_T = \sum_{t=1}^{T} \text{(Reward from chosen arm at time } t) - \sum_{t=1}^{T} \text{(Reward from optimal arm at time } t)$

Total regret measures the cumulative opportunity loss incurred by an algorithm compared to an optimal strategy over a total time horizon T. It sums the differences between the rewards obtained from the chosen arms and the rewards that could have been obtained if the algorithm always chose the arm with the highest expected reward at each time step.

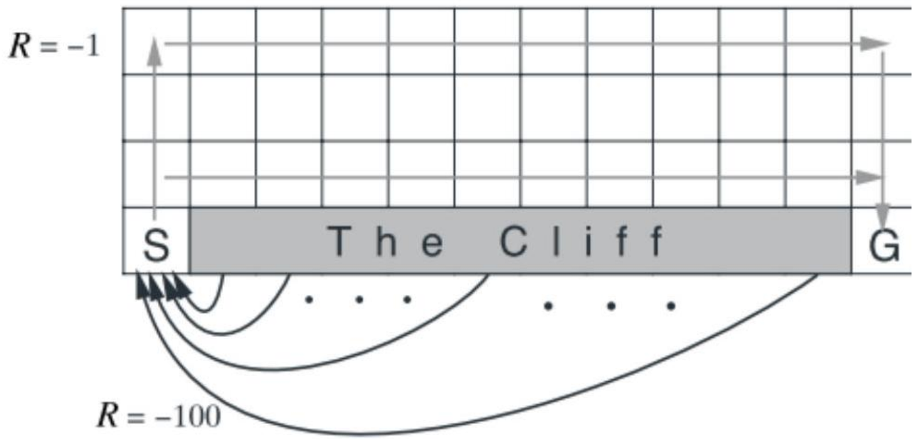
As shown in the graph, TS and UCB initially have similar average regret. However, as the number of iterations increases, TS begins to outperform UCB, if c = 1 or c = 2. This is because TS is able to exploit its knowledge of the reward probabilities to choose actions that are likely to lead to high rewards. However, if c = 0.5, it UCB outperforms TS.



3 Reinforcement Learning: Cliff Walking (40%)

Consider the cliff-walking example (Sutton & Barto, ex. 6.6. p.108). Assume that the grid has 21 columns and 3 rows (above or in addition to the cliff). This is a standard undiscounted, episodic task, with start (S) and goal (G) states, and the usual actions causing movement up, down, right, and left. Reward is -1 on all transitions except:

- the transition to the terminal goal state (G) which has an associated reward of $+20$;
- transitions into the region marked The Cliff. Stepping into this region incurs a "reward" of -100 and also terminates the episode.



safe path
optimal path

Questions

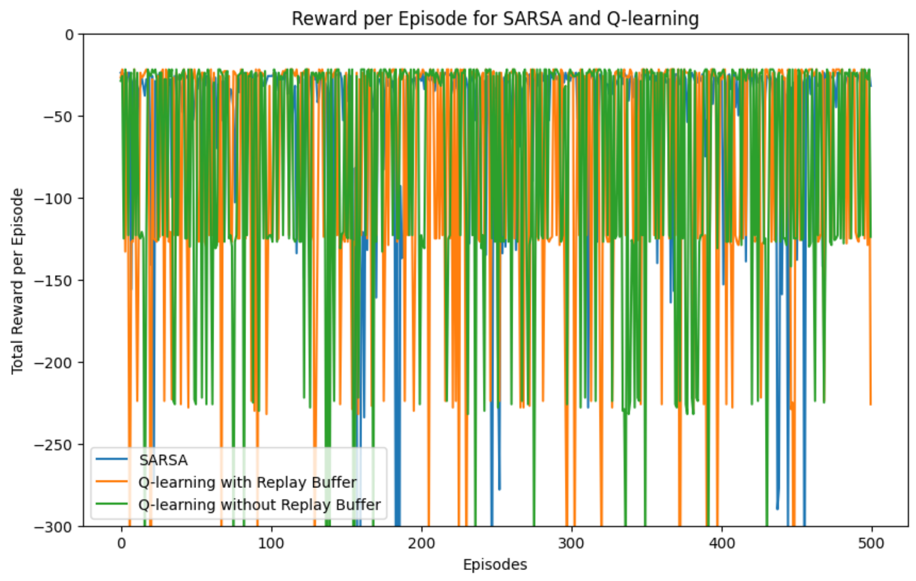
1. Use both SARSA and Q-Learning to construct an appropriate policy. Do you observe the difference between the SARSA and Q-learning policies mentioned in the text (safe versus optimal path)? For Q-learning, experiment with a replay-buffer. Discuss.

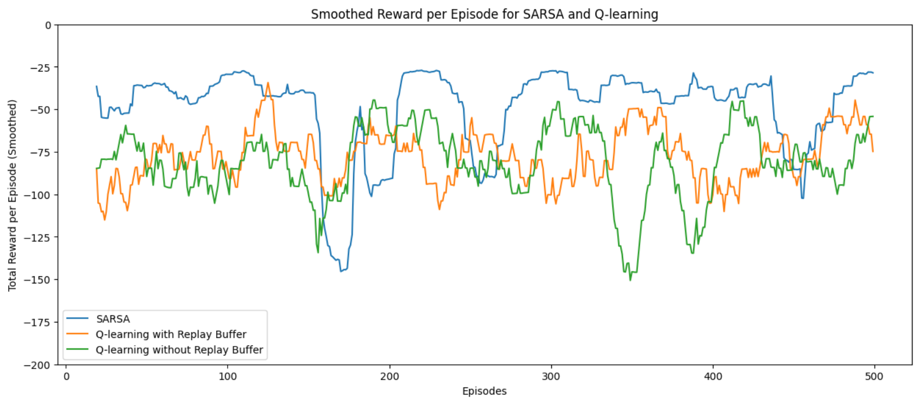
Answers

For this, I have smoothed the graph with window size 20. Otherwise, it would be too chaotic. There appears to be no much difference. SARSA appears to be slightly better, however its downward peaks are much lower. With replay-buffer is slightly better. The differences in the peaks of negative rewards between SARSA and Q-learning, as observed in the graph, can be attributed to the distinct learning behaviors of these algorithms.

SARSA, being an on-policy learning method, updates its Q-values based on the current policy, meaning it selects its next action according to its policy and then updates its Q-values accordingly. This cautious approach of SARSA tends to prioritize safer actions, resulting in a more conservative policy. Consequently, SARSA might encounter fewer negative rewards compared to Q-learning. Which can also be seen in our graph (even though its negative peaks are sometimes lower).

On the other hand, Q-learning, an off-policy learning method, aims to learn the maximum expected future reward for each action without necessarily following the current policy. This approach allows Q-learning to explore a wider range of actions, occasionally leading to riskier decisions that might result in much more peaks of negative rewards compared to SARSA. Moreover, the utilization of a replay buffer in Q-learning can influence its learning dynamics by stabilizing the learning process. This stabilization can potentially lead to a smoother overall learning curve compared to SARSA. However, despite this smoothing effect, there remains still a lot of peaks.

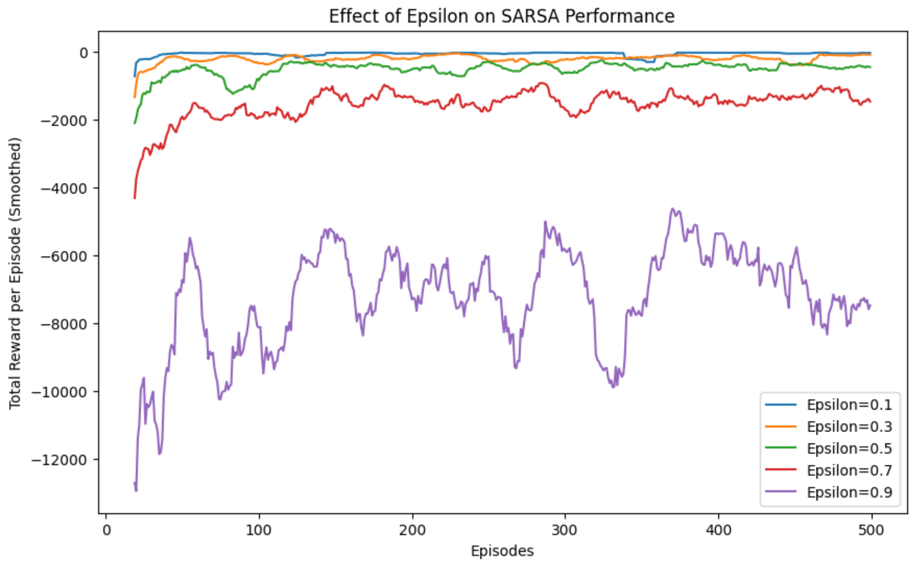


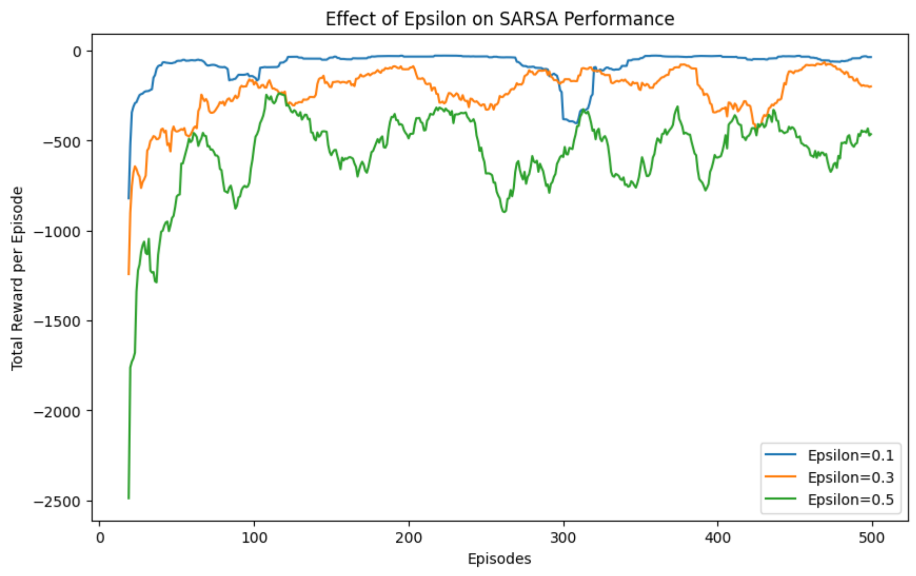


2. Try different values for ϵ (parameter for ϵ-greedy policy). How does the value of ϵ influence the result? Discuss.

Answers

The graph shows that a lower epsilon appears to be better during training. Lower epsilon values in SARSA (0.1 and 0.3) emphasize exploitation over exploration. This means the agent tends to lean more towards exploiting its current knowledge based on learned Q-values. Consequently, lower exploration leads to faster convergence toward the optimal policy, as the agent focuses on actions it believes will yield higher rewards, efficiently leveraging its learned information. We can also see this in the graph. Higher epsilon values (0.7 and 0.9), on the other hand, encourage more exploration, potentially introducing additional noise during training. While useful for initial exploration to discover new states or paths, excessive exploration clearly hindered the agent's ability to focus on the optimal path, and as can be seen slowed convergence. Also, note that for this I also smoothed the values, else it would have been too chaotic. I have also plotted epsilon for 0.1, 0.3 and 0.5 again in another graph, since this gives a better overview of the trend.

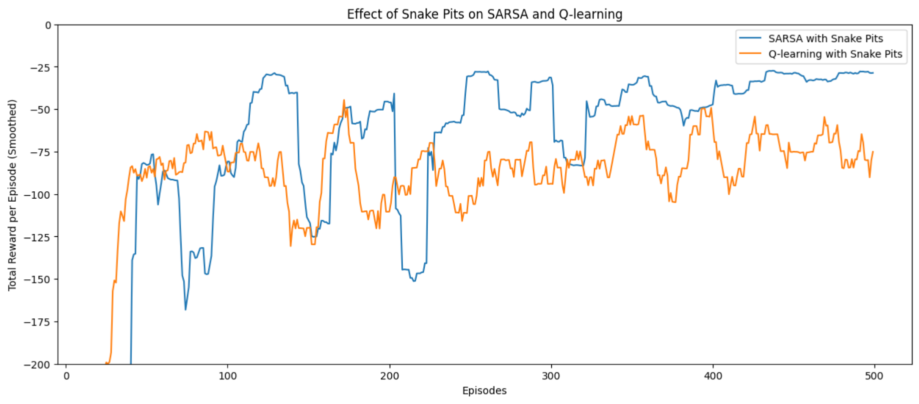




3. Now assume that there is a snake pit in the two bottom cells in the 11th column, i.e. the two cells closest to the cliff). Stumbling in these snake pits also carries a penalty (negative reward) of -100 . Does this change the results obtained by SARSA or Q-learning? Discuss.

Answers

I don't see any significant differences. SARSA still appears to be the better choice for the long run.


Effect of Snake Pits on SARSA and Q-learning
SARSA with Snake Pits
Q-learning with Snake Pits
Total Reward per Episode (Smoothed)
Episodes