RL Lab 1 - Exploration vs Exploitation

Adonis Jamal

1 Greedy Agent

1.1 How did our agent do?

Comment on the average reward graph throughout the agent's training. Is it possible for it to do better? If so, how?

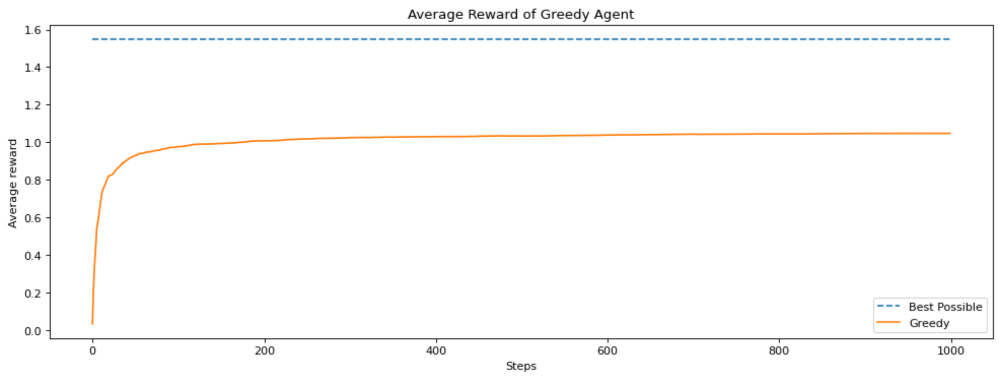

Figure 1: Average reward during training for the greedy agent

Compared to the best possible average reward, our greedy agent does not perform optimally. This is due to the greedy approach, which can cause the agent to get stuck in a local optimum and miss out on potentially better actions.

It is possible for the agent to do better by incorporating an exploration strategy, such as epsilon-greedy, which allows the agent to occasionally choose random actions and explore the action space more effectively. This can help the agent discover better actions that it might have missed with a purely greedy approach.

2 Epsilon-Greedy Agent

2.1 What do you notice?

Explain the difference between the greedy and epsilon-greedy agents and comment on the average reward graph throughout the agent's training.

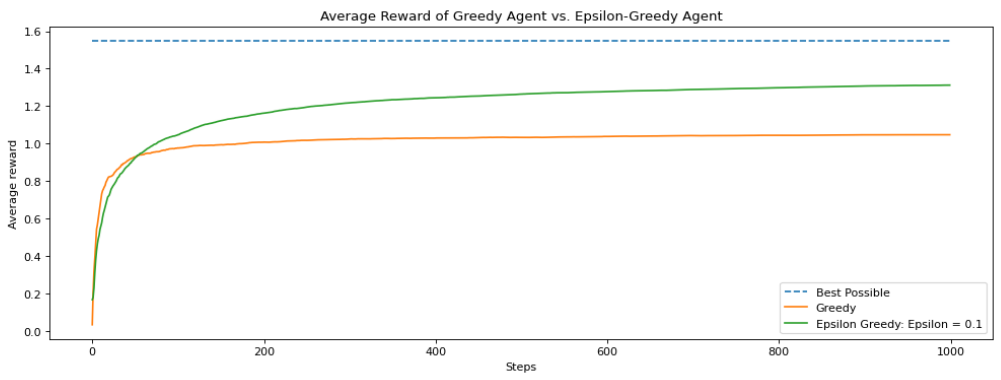

Figure 2: Average reward during training for the epsilon-greedy agent

The epsilon-greedy agent performs better than the greedy agent. The greedy agent gets stuck in a local optimum and fails to discover the true best action, while the epsilon-greedy agent explores the action space more effectively, even after finding a good action, and is able to find better actions, leading to higher average rewards.

The choice of epsilon is also important, as it represents the trade-off between exploration and exploitation. A higher epsilon encourages more exploration, which can help the agent discover better actions, but it can also lead to suboptimal performance if it explores too much. A lower epsilon encourages more exploitation, which can lead to better performance if the agent has already found a good action, but it can also lead to suboptimal performance if the agent gets stuck in a local optimum.

For the above run with epsilon = 0.1, the agent does not find the optimal action within the 1000 steps, but it does find a better action than the greedy agent, which is reflected in the higher average reward.

3 Comparing Values of ϵ

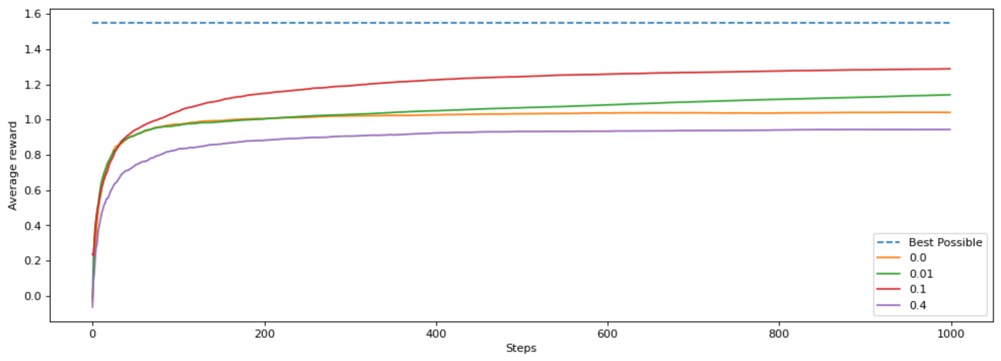

Figure 3: Comparison of average reward for $\epsilon = \{0.0, 0.01, 0.1, 0.4\}$

3.1 Why did $\epsilon = 0.1$ perform better than $\epsilon = 0.01$?

The epsilon value of 0.1 performed better than 0.01 because it provided, as discussed above, a better balance between exploration and exploitation. With an epsilon of 0.01, the agent explores very rarely, especially compared to an epsilon of 0.1, which explores other actions 10% of the time.

This means that with an epsilon of 0.01, the agent is more likely to get stuck in a local optimum and miss out on discovering better actions, while with an epsilon of 0.1, the agent has a better chance of finding the optimal action.

3.2 Why did $\epsilon = 0.4$ perform worse than $\epsilon = 0.0$ (the greedy agent)?

The epsilon value of 0.4 performed worse than 0.0 (the greedy agent) because it explores too much, which can lead to suboptimal performance. With an epsilon of 0.4, the agent explores other actions 40% of the time, which means that it is not exploiting the best-known action enough and is instead taking random actions too frequently. This can lead to lower average rewards compared to the greedy agent, which always exploits the best action it has found.

4 Effects of the Step Size

4.1 Effect of using a step size of $1/N(A)$

The agent with a step size of $1/N(A)$ performed better at the start but worse when the environment changed. Explain what happened.

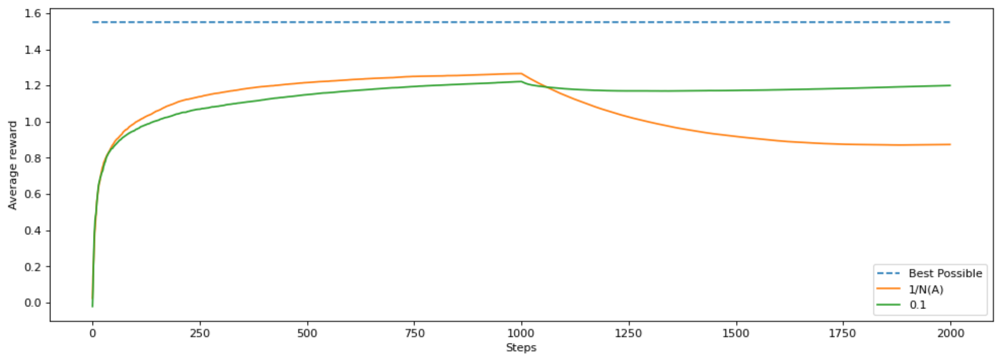

Figure 4: Effect of step size on average reward when the environment changes

Using an adaptive step size like 1/N(A) allows the agent to quickly learn the true value of an action when it is first being explored, but as the action is selected more frequently, the step size decreases, which can make it difficult for the agent to adapt to changes in the environment. In a non-stationary environment where the expected rewards of actions can change over time, a constant step size can allow the agent to continue learning and adapting to changes, while an adaptive step size that decreases over time may struggle to keep up with changes in the environment, leading to suboptimal performance.