Reinforcement Learning for Bioengineers

Coursework 1

BIOE70077

Trevi Yek - CID02145518

Autumn Term Coursework

IMPERIAL

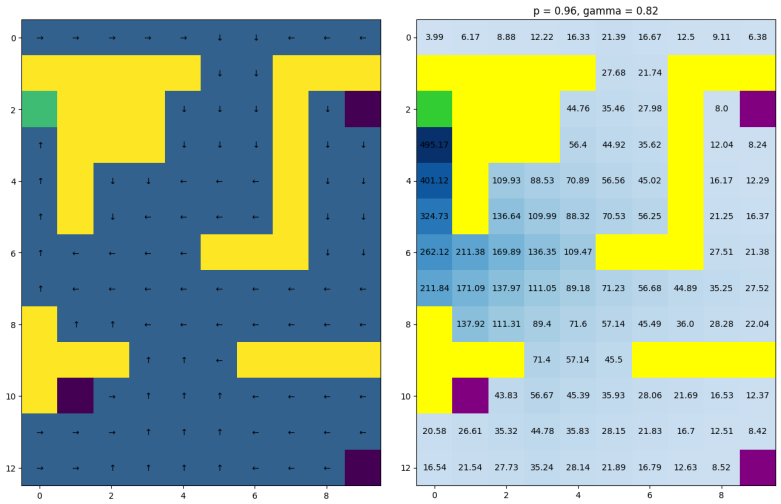
November 14, 2024

1 Dynamic programming (DP)

1.1 Method and Parameter Justification

There are two assumptions that a problem must have in order for DP to be applicable: optimal substructure and overlapping sub-problems, both which are features of a MDP. These conditions are met because the reward is only dependent on the current state and action taken at that state. The maze is small, so the difference in computational cost of Policy Iteration and Value Iteration (more expensive) will be insignificant. So I have chosen to go with the Value Iteration, due to the simplicity of the algorithm. The probability p and gamma γ has been predetermined by my CID. ($p = 0.96$, $\gamma = 0.82$)

1.2 Graphical Representation



(a) Optimal policy (b) Heat map

Figure 1.1: Result of the DP agent ($p = 0.96$, $\gamma = 0.82$)

1.3 Effect of gamma (γ) and p

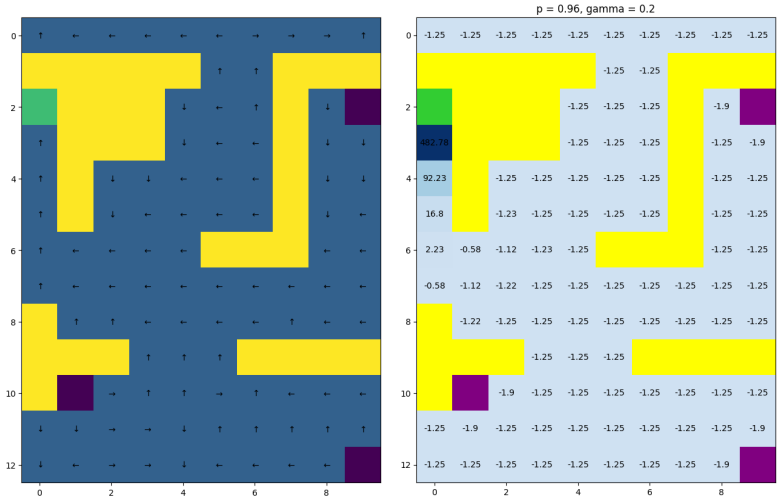


Figure 1.2: Result for $\gamma < 0.5$ ($p = 0.96$, $\gamma = 0.2$)

The discount factor γ, which ranges from 0 to 1, determines how much future rewards are valued compared to immediate rewards. By comparing Figure 1.2 ($\gamma < 0.5$) to Figure 1.1 ($\gamma > 0.5$), we can

observe that a low γ makes the agent more "short-sighted", leading to lower and sharper drop in state values as the distance from the goal increases. The optimal policy will favour paths closer to the goal, so it becomes suboptimal for states further away. As γ increases, the epochs required for convergence increases.

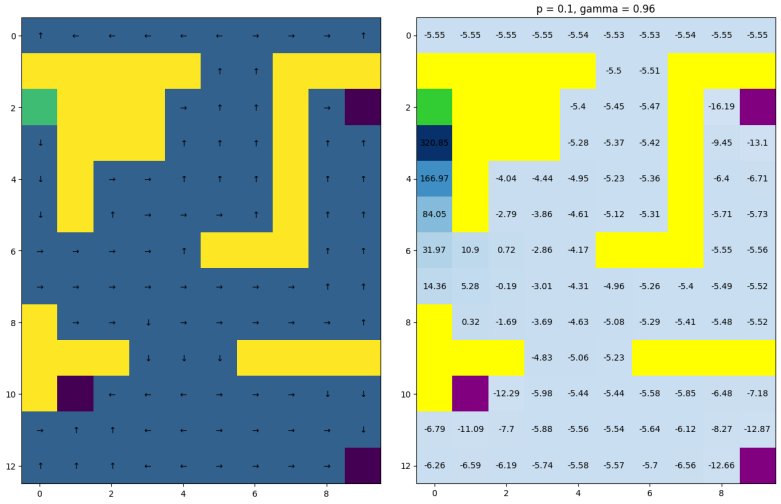


Figure 1.3: Result for $p < 0.25$ ($p = 0.1$, $\gamma = 0.96$)

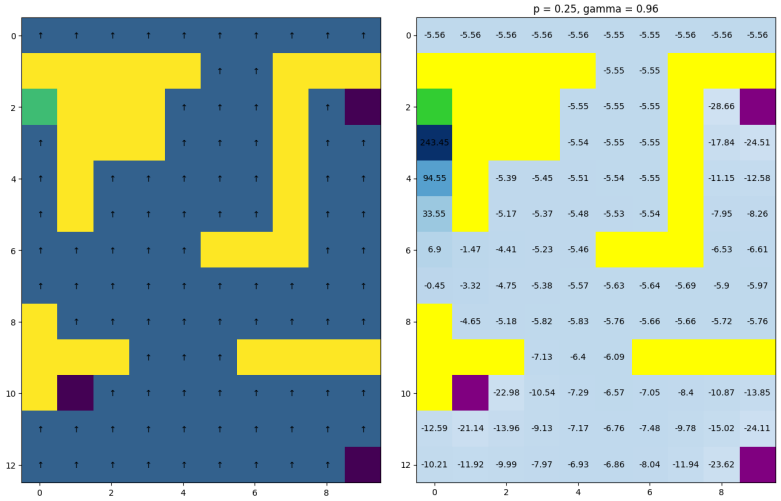


Figure 1.4: Result for $p = 0.25$ ($p = 0.25$, $\gamma = 0.96$)

The success probability is the likelihood that the agent's actions will succeed as intended. When $p < 0.25$ in Figure 1.3, the arrows in the resulting policy always points in the wrong direction. The value function has a sharp drop from the goal and becomes less distinct further out because the probability of reaching the goal from any state becomes uncertain. When $p = 0.25$ in Figure 1.4, all arrows in the optimal policy point upwards because policies does not affect actions taken. For $p > 0.25$, the resulting policy looks the same as shown in Figure 1.1. However as p increases to 1, the drop in values from the goal becomes more shallow and epochs required to converge decreases.

1.4 Extended Task

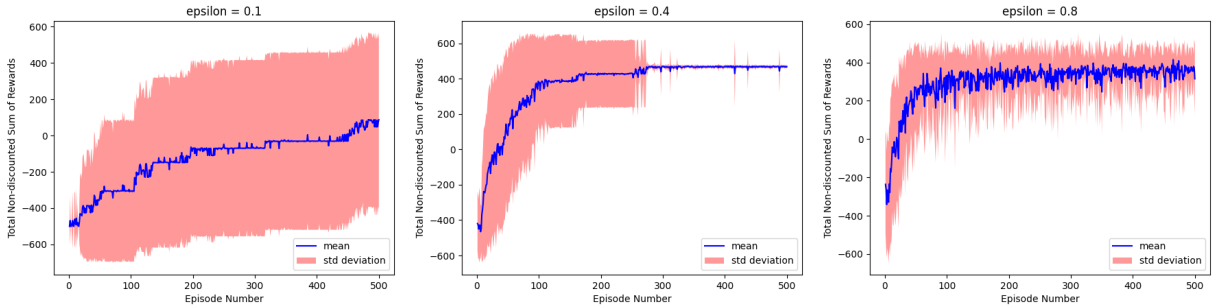
The discount factor is $\gamma = 1$, so that values can represent long-term probabilities without discounting over time to achieve "the probability of eventually reaching the goal state". Reward for the goal state = 1, reward for other terminal states = 0 and reward for an action in the environment = 0, giving probability values in the range of 0 and 1.

2 Monte-Carlo Reinforcement Learning (MC)

2.1 Method and Parameter Justification

I decided to use a first-visit Monte Carlo with decaying epsilon-greedy policy, with online update (after each episode). First-visit evaluation because it is an unbiased estimator of true state values because each state-action values used in the different episodes have independent and uniform distribution. Decaying greedy-epsilon policy for balancing out exploration and exploitation, where the agent explores the state-action space broadly in the beginning before reducing the randomness in its action, stabilising into an optimal policy.

The starting epsilon is $\epsilon = 1.0$ for fast learning speed (Figure 2.1), with a decay constant of 0.9995 and it runs for 5000 episodes. These parameters allow for appropriate amount of exploration before exploitation kicks in, to get a reliable optimal policy and value function. In Figure 2.1, low epsilon causes slow convergence into suboptimal policies and high epsilon results in high variability upon quick convergence,so having a starting epsilon of $\epsilon = 1$ which decays into approximately $\epsilon = 0.08$ after 5000 episodes uses the exploration and exploitation strategies to their advantage. For more efficiency, I have also decided to go with an incremental implementation, on an episode by episode basis, for updating the Q values.



(a) epsilon = 0.1 (b) epsilon = 0.4 (c) epsilon = 0.8

Figure 2.1: Testing MC agent with constant epsilon

These values have been chosen while considering the parameters for optimal efficiency. Decreasing the time to convergence by using a lower initial epsilon with a high decay rate can reduce the reliability (or accuracy) of the value function.

2.2 Graphical Representation

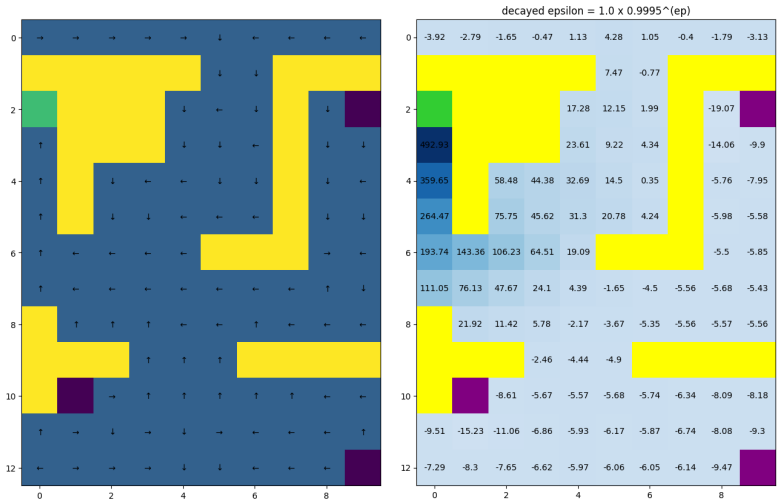


Figure 2.2: Result from MC agent with decaying-epsilon ($\epsilon_{decay} = 1.0 \times 0.9995^{ep}$, $\gamma = 0.96$)

2.3 Plot the learning curve

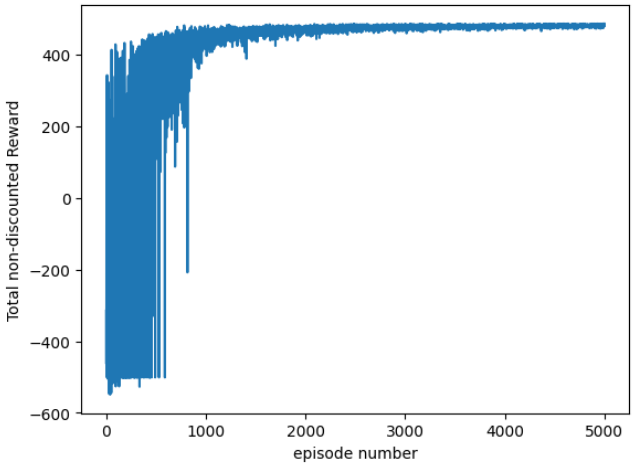


Figure 2.3: Total non-discounted rewards against episode ($\epsilon_{decay} = 1.0 \times 0.995^{ep}$, $\gamma = 0.96$)

2.4 Mean and Standard deviation across 25 training runs

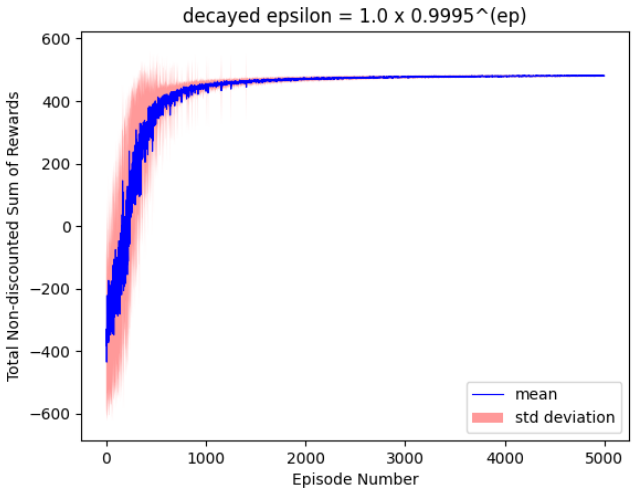


Figure 2.4: MC learning curve across 25 runs ($\epsilon_{decay} = 1.0 \times 0.9995^{ep}$, $\gamma = 0.96$)

2.5 Extended Task

Firstly, there will be a bias towards the states closer to the terminal states. So state values further away from the terminal state will be underestimated or incomplete, making them unreliable. Ignoring episodes over 500 steps could reduce the variance in the return estimates for the closer states, because more of the sampled episodes would focus on nearer states. This could help MC estimates converge faster for the closer states because they get more frequent updates.

3 Self-directed Research and Comparative Analysis

3.1 Exploration Strategy Comparison - Parameters description

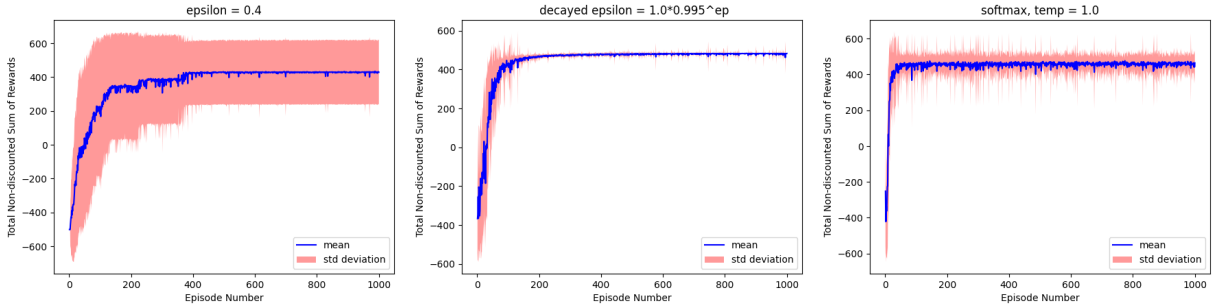
The class is named "MC_agent_Q3" and I have selected specific parameters for running the next section. The epsilon value of 0.1 is too low to produce usable outcome (the value function only contains negatives and zeros). So I went with the previous epsilon values I used and a slightly higher decay constant, because I want to compare the methods after some fine-tuning to try to get the best results (ie to allow each method to perform its best for analysis).

For the fixed Epsilon, $\epsilon = 0.4$. For the decaying Epsilon, starting $\epsilon = 1.0$ with decay constant of $k = 0.995$ ($\epsilon_{decayed} = \epsilon \times k^{episode}$). And temperature is set to 1.0 for the Softmax (better known as Boltzmann Exploration) approach.

3.2 Analysis Requirements: Learning Curves, Final Policy and Value Function

By comparing the learning curves in Figure 3.1, we are able to observe the distinct difference caused by the balance of exploration and exploitation for each strategy.

Figure 3.1(a), fixed epsilon, displays significant variability (standard deviation) throughout training due to the continuing exploration. Figure 3.1(b), decaying epsilon, displays significant variability in the beginning and then converges with the mean. Figure 3.1(c), Softmax display a fast convergence and a reasonable amount of exploration after convergence.

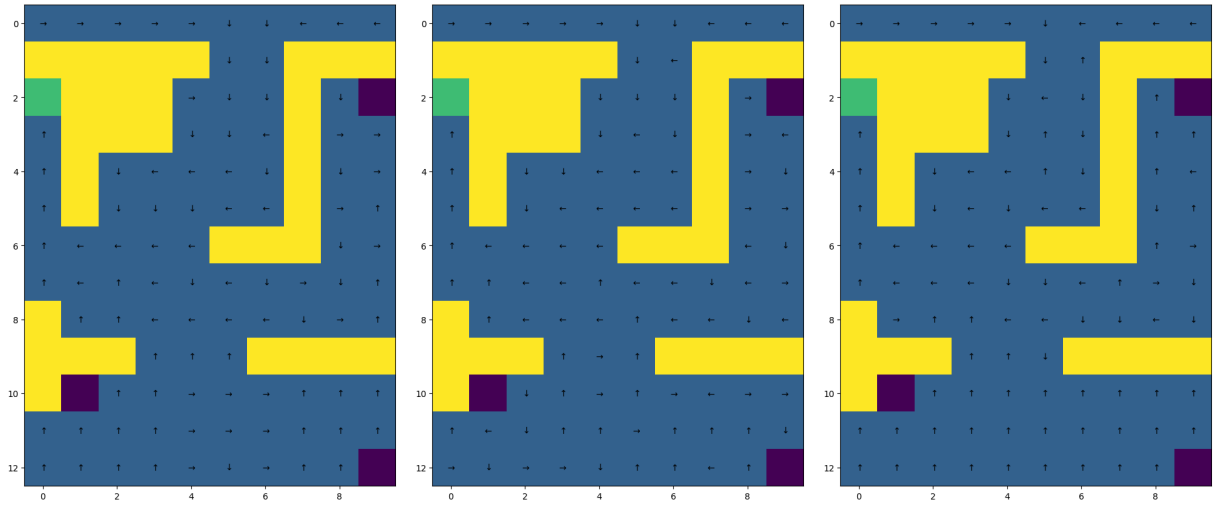


(a) fixed epsilon (b) decaying epsilon (c) softmax

Figure 3.1: Learning curve for each strategy

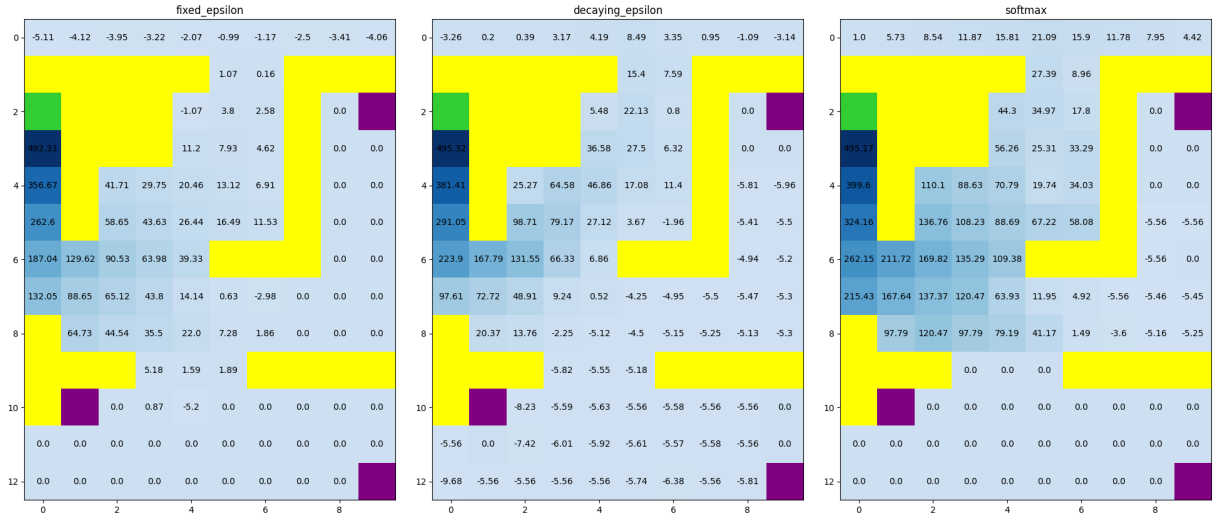
In Figure 3.2, the policies are quite accurate in the regions of the starting states and the goal (reward state). However, they are unreliable in the south region of the map, where there ware been less explorations.

The heat maps in Figure 3.3 displays an increasing shallowness of drop in value from the goal for fixed epsilon, decaying epsilon and Softmax, in order.



(a) fixed epsilon (b) decaying epsilon (c) Softmax

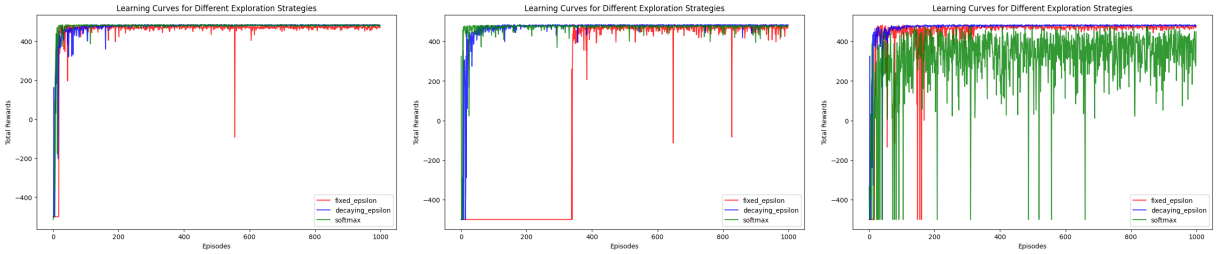
Figure 3.2: Resulting optimal policy for each strategy



(a) constant epsilon (b) decaying epsilon (c) Softmax

Figure 3.3: Resulting value functions for each strategy

This observation is reflected in Figure 3.4 where there is a significant possibility of suboptimal convergence and excessive exploration for the fixed-epsilon and Softmax strategies. The decaying-epsilon seems to be more reliable for this environment, which leads us to the next section.



(a) Good overall results (b) Weird result from fixed epsilon (c) Weird result from Softmax

Figure 3.4: Total rewards against episode number

3.3 Comparative Analysis: The Chosen Strategy and Observed Differences

Because this environment is static and moderately complex, I have decided that the decaying-epsilon strategy is the most suitable for the environment. It provides convergence after a suitable amount of exploration.

Sutton & Barto (2018) discuss the trade-offs involved in exploration versus exploitation, emphasizing the need to balance these to achieve policy convergence and optimality. Exploration is needed to identify optimal actions and exploitation is required to refine or stabilise the optimal policy, but both are mutually exclusive during implementation. This phenomenon was coined as '**the Exploration-Exploitation Dilemma**' (Sutton & Barto 2018, p. 32).

The fixed epsilon (a near greedy action selection method) allows for a stable amount of exploration throughout training, resulting in continuous adaptation. Assuming that there is no step limit and an optimal epsilon is chosen, theoretically, every action can be sampled infinitely to result in convergence. However, in practice with finite number of steps, it results in either excessive exploration or exploitation in later stages, not fully selecting optimal actions or getting stuck performing suboptimal actions. (Sutton & Barto 2018, p. 58–60). Hence fixed-epsilon may be more useful for simple, dynamic environments.

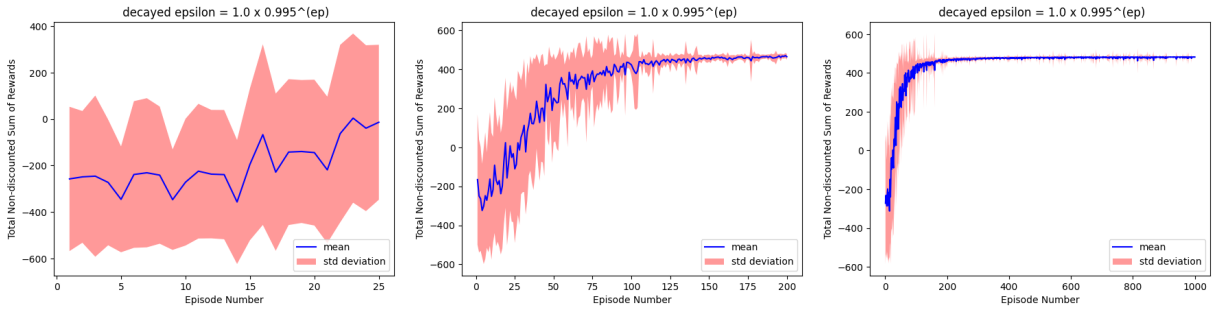
Decaying epsilon makes epsilon a function of time or step. It provides a gradual shift from exploration to exploitation, which is an improvement from the fixed epsilon method (Szepesvári 2010, p. 39).

Slower decay rate and more episodes allows for more exploration before convergence, hence the decay rate and initial epsilon has to be fine-tuned with a trade-off between accuracy and efficiency. It is 'simple and efficient' Szepesvári (2023). There is limited adaptability after convergence, so it is more suitable for complex, static environments.

Softmax (Boltzmann in the textbooks) exploration is more targeted because it takes the relative values of the actions into consideration, allowing for smoother transitions between exploration and exploitation. (Szepesvári 2010, 2023) However this method is the most computationally expensive out of the three, and a certain level of exploration will still happen after convergence, albeit less than what was observed for fixed-epsilon. The value of temperature is also harder to fine-tune compared to the straightforward epsilon greedy methods. So we can say that the better use case for this strategy is in more complex and dynamic environments.

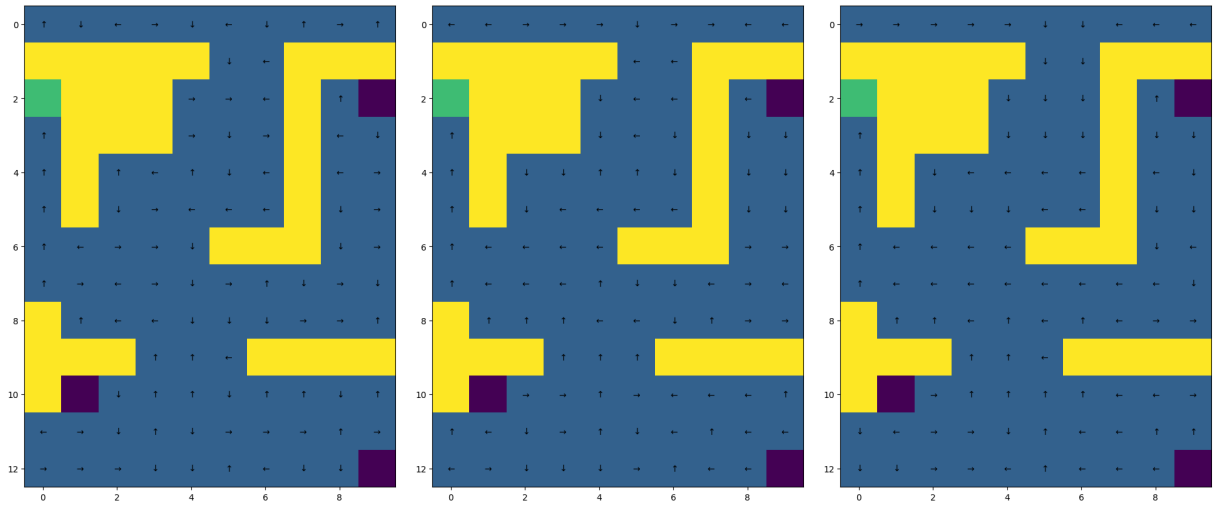
3.4 Sensitivity Analysis: Using the decayed-epsilon strategy

By observing Figure 3.5, we can conclude that higher number of episodes increases the possibility and reliability of the convergence, due to more state trajectories sampled by more episodes. This pattern is also observed in Figure 3.6 for the optimal policies, as more episodes provides more accuracy.



(a) 25 episodes (b) 200 episodes (c) 1000 episodes

Figure 3.5: Learning curve for varying number of episodes



(a) 25 episodes (b) 200 episodes (c) 1000 episodes

Figure 3.6: Output policy for varying number of episodes

References

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement Learning, Second Edition : An Introduction*, MIT Press, [ProQuest Ebook Central].
URL: *http://ebookcentral.proquest.com/lib/imperial/detail.action?docID=6260249*

Szepesvári, C. (2010), *Algorithms for Reinforcement Learning*, Springer International Publishing AG, [ProQuest Ebook Central].
URL: *http://ebookcentral.proquest.com/lib/imperial/detail.action?docID=881218*

Szepesvári, C. (2023), 'What is exploration strategies in reinforcement learning?', https://medium.com/@aiblogtech/what-is-exploration-strategies-in-reinforcement-learning-32677239245e. Accessed: 2024-11-14.