Reinforcement Learning Strategies with Multi-Armed Bandit Report

Pantelis Kasioulis, Ewa Fojcik, Gabriel Thomas Newton and Zoi Kallinaki

Problem 1 – Implementation

This implementation addresses the 10-armed bandit problem using the ε-greedy algorithm to investigate the exploration-exploitation trade-off. It evaluates how different exploration rates ($\varepsilon = 0$, $\varepsilon = 0.01$, $\varepsilon = 0.1$) affect the agent's performance in an environment with stationary reward distributions. By calculating average reward, percentage of optimal actions and cumulative regret, the algorithm provides evidence regarding the impact of various exploration strategies on decision-making under uncertainty.

Environment Design: The implementation models a bandit with 10 arms, where each arm's true value is drawn from $N(0,1)$ at initialization. These values remain fixed throughout the experiment, with rewards generated from $N(true\_value, 1)$ when an arm is selected. Figure 1 illustrates this environment through violin plots, showing the distribution of potential rewards for each arm. The black horizontal lines represent each arm's true value, with Arms 4 and 7 having the highest values around 1.5. The overlapping distributions demonstrate why the problem is challenging-even the optimal arm occasionally yields lower rewards than suboptimal arms.

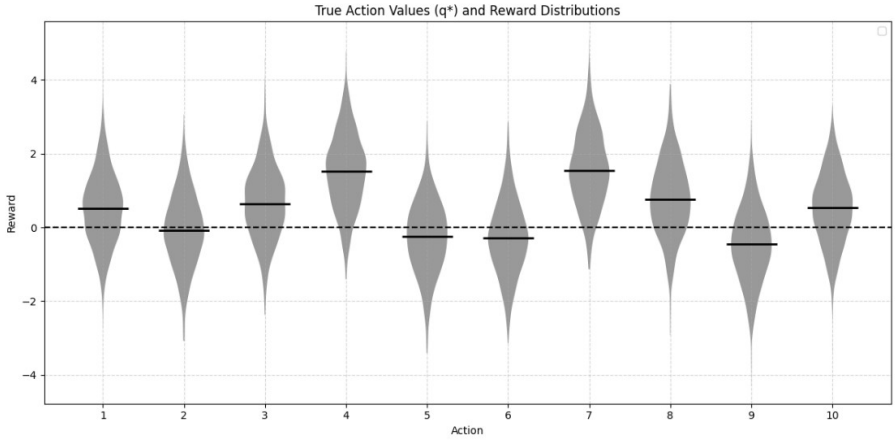

Figure 1: Sutton & Barto: Violin plot of True Action Values and Reward Distributions (similar to Figure 2.1 in Sutton & Barto)

Agent Architecture: The agent utilizes the sample-average method to estimate action values and employs an ε-greedy policy for action selection. The policy implementation includes several key mechanisms. For value estimation, action values are updated incrementally after each reward observation using the following formula (further explained in Problem 3):

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{N_t(a)}[R_t - Q_{t-1}(a)]$$

The action selection follows the ε-greedy approach where with probability $1 - \varepsilon$, the agent selects the action with the highest estimated value, while with probability ε it selects a random action. When multiple actions share the maximum estimated value, ties are broken randomly. For performance tracking, beyond basic rewards and actions, the implementation calculates instantaneous regret by comparing the true value of the selected action against the true value of the optimal action, providing precise measurement of opportunity cost (Auer & Ortner p. 55-65).

Experimental Protocol: To ensure robust results, the implementation conducts 2000 independent runs for each exploration parameter. Each run consists of 2000 sequential action selections and updates. Results from all runs are aggregated to produce average performance metrics and visualizations for comparative analysis.

Experimental Results:

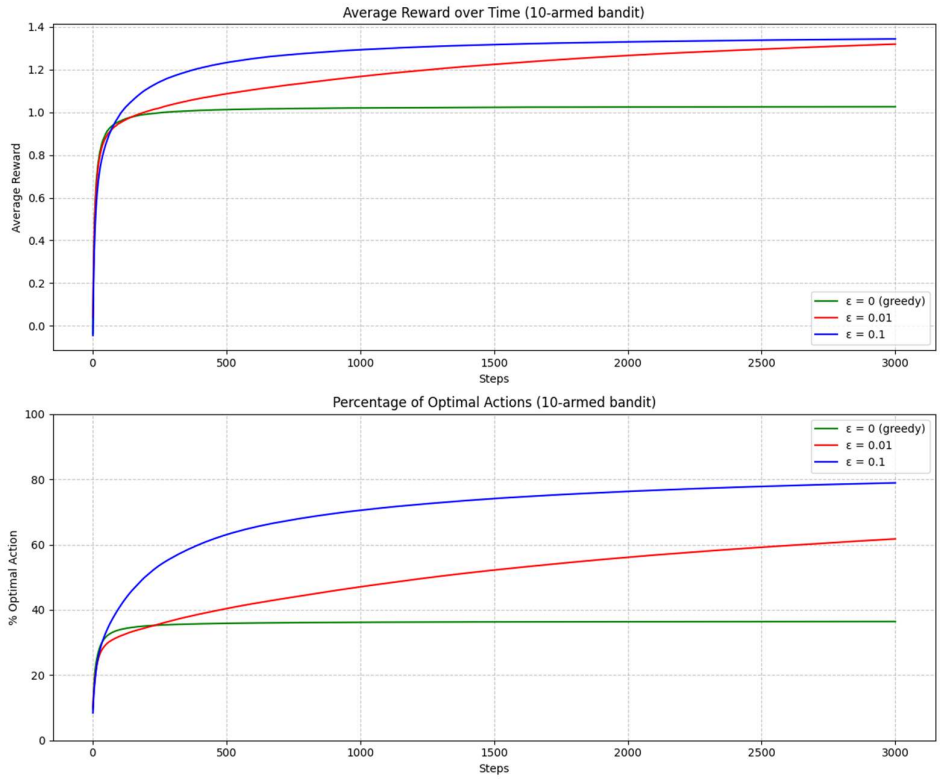

Figure 2: Average Rewards Over Time and Percentage of Optimal actions (similar to figure 2.2 in Sutton & Barto)

METRIC	$\boldsymbol{\varepsilon = 0}$ (GREEDY)	$\boldsymbol{\varepsilon = 0.01}$	$\boldsymbol{\varepsilon = 0.1}$
AVERAGE REWARD	1.0111	1.2726	1.3244
OPTIMAL ACTION SELECTION	36.06%	59.30%	76.01%
FINAL CUMULATIVE REGRET	1024.48	499.70	397.21

The purely greedy method ($\varepsilon = 0$) quickly plateaued at suboptimal performance due to its inability to correct initial misconceptions. As shown in Figure 2, its average reward levels off at around 1.0 after approximately 270 steps, while the more exploratory approaches continue to improve. The greedy method ultimately achieves only 1.0111 average reward, substantially below the other strategies. With $\varepsilon = 0.01$, performance improved markedly, showing a steady upward trajectory throughout the 2000 steps and achieving a higher average reward of 1.2726 compared to the greedy approach. The $\varepsilon = 0.1$ configuration delivered the best performance, with its average reward climbing most rapidly and reaching the highest level of 1.3244.

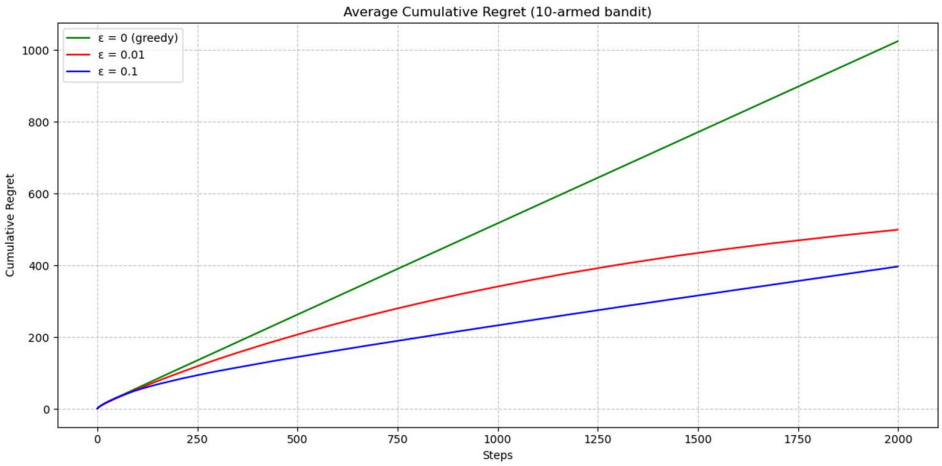

Figure 3: Average Cumulative Regret Graph

Further Observations: Interestingly, when extending the experiments beyond the standard 2000 steps, we observe that around step 3500, the $\varepsilon = 0.01$ strategy begins to outperform $\varepsilon = 0.1$. This suggests that the optimal exploration rate depends on the time horizon – higher exploration rates provide better performance in shorter horizons by accelerating the discovery of optimal actions, while more moderate exploration rates become advantageous in longer horizons by reducing the cumulative cost of unnecessary exploration once optimal actions have been identified.

Conclusion: This implementation confirms that appropriate exploration is crucial for optimal performance in reinforcement learning. The ε-greedy algorithm with $\varepsilon = 0.1$ consistently outperformed both the greedy approach and the minimal exploration configuration across all performance metrics within the 2000 step horizon, demonstrating the benefit of balancing exploration and exploitation in uncertain environments. The visualizations in Figures provide clear evidence of how different exploration strategies affect learning dynamics and overall performance in the multi-armed bandit problem.

Problem 2 – Exploration and Exploitation

Reinforcement learning relies on feedback, which provides information on how rewarding an action was, as opposed to providing direct instructions to agents on which actions are deemed correct to take. Given that the agent does not know the value of each action in the n-armed bandit problem, this pushes the need for active exploration. Therefore a trial-and-error approach is required to discover the most desirable behaviour. In order to do that, the exploration-exploitation approach is necessary.

If the action values are stationary, then at any time step there is always one or more actions whose estimated values are the highest. In the purely greedy strategy, the agent always chooses the highest (greedy) estimated value. In that case, the agent is exploiting its existing knowledge of the values of actions. If the agent were to focus just on the exploitation part, that would then maximise the expected reward on one step, as it would get the highest possible reward based on its current knowledge. This becomes useful when immediate results take priority over long-term optimization (Sutton & Barto p. 33). Pure exploitation might also be more effective in stable environments, if the reward distributions of actions are static and well-know. It is also easier to implement, as it does not require adding computations for exploration probabilities or confidence bounds (Sutton & Barto p. 41).

Despite the advantages of a greedy strategy, even though it maximises the immediate rewards in one step, exploration usually produces a greater total reward in the long run (Sutton & Barto p. 32). This is because the agent needs to try different actions to discover which ones yield the best long-term outcomes. To discover those actions however, the agent must explore the available options to make

better action selections in the future (Sutton & Barto p. 3). If the agent were to only choose one action, it would not have the chance to discover new actions that might potentially lead to even higher rewards. Lack of exploration can cause the agent to find itself stuck by repeatedly selecting only the 'safe' values that it knows to be effective. By doing that it can potentially miss out on better options that it had not tried sufficiently.

This trade-off between exploration and exploitation is one of the key challenges in reinforcement learning. Most of the sophisticated methods for balancing exploration and exploitation assume stationary and prior knowledge, which are often violated or unverifiable in real-world problems, as the need for continuous exploration is even greater in non-stationary environments (Sutton & Barto p. 33 & 36).

In conclusion, while pure exploration ensures faster convergence, efficiency and maximisation of short-term rewards, exploration ensures the agent does not settle for suboptimal choices. In the next section practical strategies, like the ε-greedy method and Upper Confidence Bound (UCB), are explored in more detail. These methods help to achieve the balance between exploration and exploitation, leading to more efficient learning in multi-armed bandit problems.

Problem 3 – Action-Value Methods

In an ideal scenario, an agent would be able to train a model that consistently selects the best combination, achieving optimal results every time. However, in real applications, this is unrealistic. In practice, when dealing with multi-armed bandits in machine learning, an agent must navigate the exploration-exploitation dilemma while interacting with an uncertain environment. This dilemma aims to optimise the results to the highest achievable percentage. This is done through action-value methods.

Action-value methods, first proposed by Thathachar and Sastry (1985), often referred to as *estimator algorithms (*Sutton & Barto p. 50), in the context of the multi-armed bandit, can be distinguished between action-value estimates and action rules (which are essentially action selection strategies) (Sutton & Barto p. 34). Both of these are crucial in reinforcement learning because, if it involves planning, it must balance decision-making and a real-time action selection, with model acquisition and improvement (Sutton & Barto p. 4).

An example of a simple method to estimate the values of actions, in order to make action selection decisions is a *sample-average* method (Sutton & Barto p. 36). This way, each estimate is derived by averaging the observed relevant rewards. It is not the only method, nor necessarily the best, but it serves as a simple baseline for understanding action-value estimation. Its simplicity also makes it useful for initial exploration and understanding of the fundamental concepts (Sutton & Barto p. 34).

Using the average-sample method in its initial form might become problematic when it comes to storing all past rewards and recalculating the average each time a new reward is received. This requires memory and computation that increases with the number of rewards (Sutton & Barto p. 37).

$$Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{N_t(a)} R_i$$

Each step would produce an R which represents the rewards found from selecting the action a selected at step i. The $N_t(a)$ represents the number of times action a has been chosen up to time step t. The values are slowly collected and the $Q_t(a)$ gradually converges to the true value.

To avoid this issue, a simple solution would be to implement incremental update formula:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{N_t(a)}[R_t - Q_{t-1}(a)]$$

In the updated formula $Q_{t-1}(a)$ represents the previous estimate of the action value, R_t is the new observed reward, and $N_t(a)$ is the updated count of times action α has been selected. By implementing this into the average-sample method, instead of storing all past rewards, the agent only needs to store the current estimated value, the number of times the action has been taken and the most recent reward (Sutton & Barto p. 37). This method saves computational expense and ensures more stable estimates.

In the Python code used in Problem 1, Sutton's (2014) guidance's been followed by implementing the ε-greedy. The ε-greedy method balances exploration-exploitation in reinforcement learning. It selects the best-known action using the probability formula of $(1 - \varepsilon)$ and then explores randomly using the probability ε. For example, $\varepsilon = 0.1$, the agent would essentially explore 10% of the time and exploit 90%, ensuring continuous learning and convergence toward optimal actions.

Another action selection strategy example is the *upper confidence bound (UCB)*. This strategy selects actions based on their estimated value and the uncertainty in that estimate. This approach pushes the agent to try actions that have not been tried as often as others and that also have a possibility of scoring higher than the greedy action. Their uncertainty makes them potentially more rewarding than the greedy action, which is the factor driving exploration (Sutton & Barto p. 41). While effective in bandit problems, the use of UCB's might be however limited in more complex reinforcement learning scenarios, especially in those involving nonstationary environments or large state spaces with function approximation (Sutton & Barto p. 42).

Notably, all action value selection methods, such as UCB, greedy and ε-greedy, are partially influenced by the initial assumptions about action values – which results in bias. The bias could be minimised using a dynamic ε-greedy, or by implementing optimistic initial value (Sutton & Barto p. 40). These methods encourage exploration, particularly in the early stages of learning, but they also come with their own limitations and, due to assignment constraints, were not adopted in this study's solution for Problem 1.

*****All problems, including the implementation of code were worked on collaboratively, reviewed and edited by all group members.*

References:

Auer, P., & Ortner, R. (2010). UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, *61*(1-2), 55-65.

Sutton, H. "Peter Morgan Sutton." *BMJ*, vol. 348, no. mar31 11, 2014, p. g2466, https://doi.org/10.1136/bmj.g2466.

Thathachar, M. A. L., and P. S. Sastry. "A New Approach to the Design of Reinforcement Schemes for Learning Automata." *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, 1985, pp. 168–175.