Question 1: Implement a multi-armed bandit algorithm

In this study, we built a multi-armed bandit simulation to assess how well the ϵ-greedy algorithm performs across different levels of complexity. The environment included n = 5, 10, and 20 arms. We ran 2,000 independent simulations, each lasting 2,000 steps.

We analysed the results using two main metrics: **Average Reward** and **Percentage of Optimal Action**. As shown in Figure 1, all settings started at zero, but differences became clear as learning progressed. The agent with (ϵ = 0.1) showed steady improvement and eventually achieved higher average rewards than both the greedy agent (ϵ = 0) and the more cautious agent (ϵ = 0.01). This pattern was also displayed in Figure 2. The ϵ = 0.1 agent found and chose the best arm faster and more frequently, while the greedy agent often got stuck at a lower success rate.

The greedy approach(ϵ = 0) may earn quick rewards early by exploiting what seems best, but it can easily lock onto an arm that is not the best option due to early estimation errors or noise. Since it never explores, it cannot correct these mistakes later. On the other hand, the ϵ = 0.1 strategy spends some time exploring at first, but this exploration helps it achieve much better long-term performance. When the number of arms increased to 20, the results also showed that a larger search space makes exploration even more important for correctly identifying the best arms.

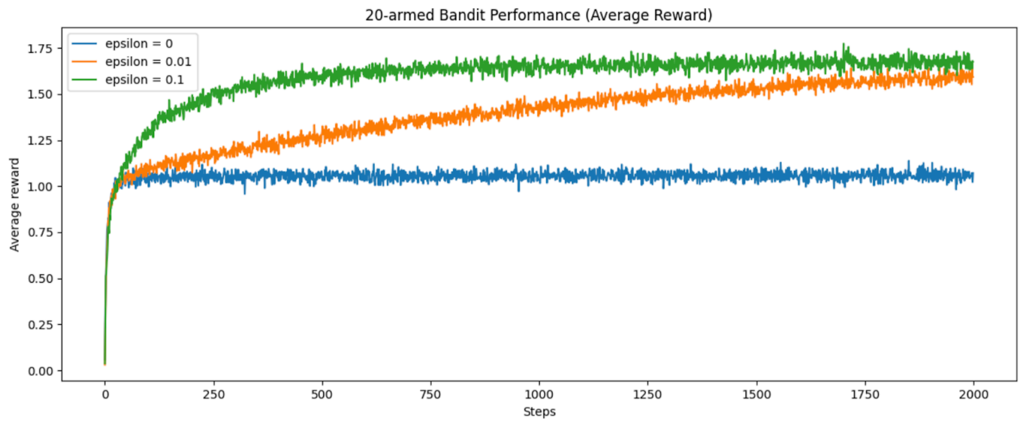

Figure 1: Average Reward performance of ϵ-greedy algorithms.

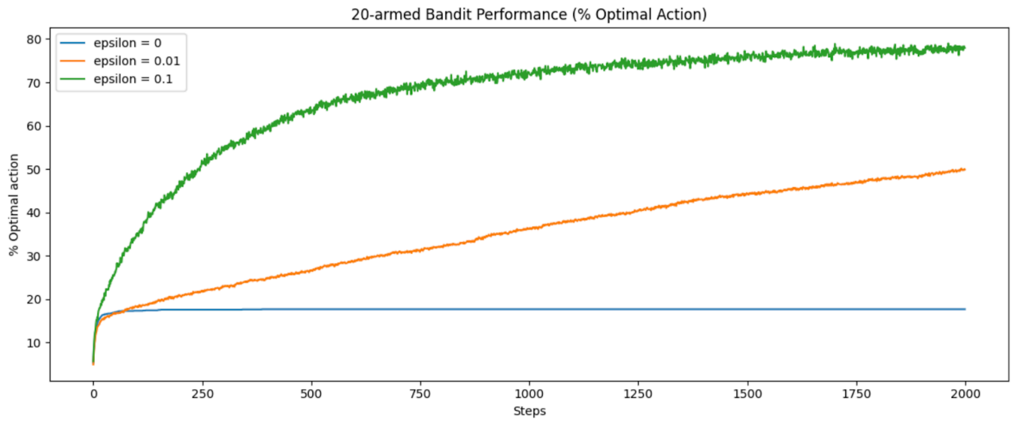

Figure 2: Percentage of optimal action selection over time.

Question 2: Explain exploration and exploitation for multi-armed bandits.

1. The Dilemma: A fundamental challenge in reinforcement learning is the trade-off between Exploration and Exploitation. An agent must balance these two competing needs to maximize cumulative rewards over time.

2. Exploration: Exploration means trying actions that don't seem the best at the moment. The aim is to learn more about the environment and maybe find better choices for the future. This is like trying a newly opened restaurant. There is a risk that the food might be terrible (low reward), but there is also a chance it could be the best meal you've ever had (discovering the optimal action).

    - Purpose: To discover potentially better actions that could lead to higher long-term rewards.
    - Risk: In the short term, exploration may lead to lower rewards because the agent is not choosing the best-known option.

    In our code, this is triggered by the `epsilon` probability, leading the agent to `np.random.randint(n_arms)` to try a random arm.

3. Exploitation: Exploitation means picking the action that seems best based on what the agent has learned so far. The aim is to get the biggest reward right now by using what it already knows. This is like going to your favorite restaurant and ordering your "usual" dish. You are certain it will be delicious.

    - Purpose: To maximize the immediate reward by using the best-known knowledge.
    - Risk: If the agent only exploits, it might get stuck with a sub-optimal action because it never tried other possibilities that could have been better.

    In our code, this happens when `np.random.random() > epsilon`, where the agent chooses the action with the highest `q_estimates`.

4. ϵ-greedy Strategy: We use the ϵ-greedy method to balance these two. The agent chooses the best-known action most of the time (with a probability of $1-\epsilon$) and sometimes tries a random one (with a probability of ϵ).

5. Experimental Observation: Based on the simulation results in Problem 1, we balanced these two behaviors using the epsilon-greedy strategy. By comparing different values of ϵ (0, 0.01, and 0.1), we observed the following:

    - $\epsilon = 0$ (Pure Greedy): Gained rewards quickly at the start, but later did worse because it couldn't find the best option.
    - $\epsilon = 0.1$: Achieved the highest average reward in the long run and identified the optimal action more quickly.
    - $\epsilon = 0.01$: Improved more slowly than $\epsilon = 0.1$, but its performance eventually surpassed the greedy strategy.

Question 3: Action-Value Methods

1. Definitions:
   The use of Action-value methods is to estimate the values of different actions and use these estimates to make decisions.

   - True Value: Denoted as $q_*(a)$, it is the actual mean reward of action a.
   - Estimated Value: Denoted as Qt(a), it is the estimate of $q_*(a)$ at time step t. Our goal is to make Qt(a) as close to $q_*(a)$ as possible.

2. Incremental Implementation: To improve computational efficiency and save memory, we use an incremental update formula to calculate these averages without storing the entire history of rewards
   Update Rule:

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

NewEstimate = OldEstimate + StepSize [Target − OldEstimate]

   - **Qn**: The old estimate for the action value.
   - **Rn**: The reward received for the n-th trial.
   - **[R(n) - Q(n)]**: This represents the prediction error, the difference between the actual reward received, R(n), and the current estimate, Q(n).
   - **1/n**: The step-size (learning rate). As n increases, the impact of the latest reward on the average decreases.

   This demonstrates how the agent refines its value estimates by adjusting the old estimate toward the target reward, using the prediction error [R(n) - Q(n)] to correct its expectations over time.

3. Decision Making
   - **Greedy**(Exploitation): Always selects the action with the highest estimated value to maximize immediate reward.
   - **ϵ-greedy**(Exploration): Most of the time selects the greedy action, but with a small probability ϵ, it selects a random action to discover better possibilities.