Experiment 7
Implementing MENACE and Epsilon-Greedy Algorithms for Stationary and Non-Stationary Bandit Problems

Tejas Pakhle, Rajat Kumar Thakur, Tanay Patel, Abhi Tundiya,

***Abstract*—This assignment explores the implementation of the Matchbox Educable Noughts and Crosses Engine (MENACE), a simple reinforcement learning system developed by Donald Michie. In addition, the epsilon-greedy algorithm, a fundamental approach for balancing exploration and exploitation in reinforcement learning, is applied to both binary and 10-armed bandit problems. The binary bandits provide stationary rewards, while the 10-armed bandit introduces non-stationary rewards, where mean-rewards undergo independent random walks. A modified epsilon-greedy agent is developed to handle the non-stationary nature of the 10-armed bandit.**

I. Implementing MENACE for Tic-Tac-Toe

1. State Space (S)

The state space represents all possible configurations of the Tic-Tac-Toe board. A board state is represented as a string of nine characters, where each character can be:

- X: Represents a move made by the AI (MENACE).
- O: Represents a move made by the human player.
- : Represents an empty space on the board.

Let $s \in S$ be a state defined as:

$$s = [s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9]$$

where each $s_i \in \{X, O, \}$ for $i = 1, 2, \ldots, 9$.

The total number of unique board configurations can be computed as:

$$|S| \leq 3^9 = 19,683$$

2. Action Space (A)

The action space defines the possible moves that can be made from a given state. Each action corresponds to placing a marker in one of the empty spaces on the board. Let $a \in A$ be an action defined as:

$$a = \{i | s_i = \text{ for } i = 1, 2, \ldots, 9\}$$

where a is the set of indices representing valid moves (the empty spaces).

The size of the action space $A(s)$ from state s can be defined as:

$$|A(s)| = \sum_{i=1}^{9} I(s_i =)$$

where I is an indicator function that returns 1 if s_i is empty and 0 otherwise.

3. Transition Function (T)

The transition function defines the result of taking an action a in state s, leading to a new state s':

$$s' = T(s, a)$$

The new state s' is formed by replacing the empty space at index a in s with the current player's marker.

4. Reward Function (R)

The reward function assigns a numerical value based on the outcome of the game:

- $R(s, a) = +1$ if the player wins after the move.
- $R(s, a) = 0$ if the game is drawn.
- $R(s, a) = -1$ if the player loses after the move.

II. Binary Bandit Problem

Problem 2: Binary Bandit Problem with Epsilon-Greedy Algorithm

State Space

The state space for the binary bandit problem can be defined as follows:

$$S = \{s_1, s_2\}$$

where:

- s_1: represents the state when the first action is selected.
- s_2: represents the state when the second action is selected.

Action Space

The action space consists of two actions:

$$A = \{a_1, a_2\}$$

where:

- a_1: represents selecting the first action.
- a_2: represents selecting the second action.

Epsilon-Greedy Algorithm

The **epsilon-greedy algorithm** is a strategy used in reinforcement learning to balance exploration and exploitation:

- With probability $1 - \epsilon$ (in this case, 0.9), the algorithm chooses the action with the highest estimated reward:

$$A = \arg\max_{a \in A} Q(a)$$

- With probability ϵ (in this case, 0.1), the algorithm randomly selects one of the available actions:

$$A = \text{random selection from } A$$

- The algorithm updates the Q-values based on the received reward R:

$$Q(A) = Q(A) + \frac{R - Q(A)}{N(A)}$$

where $N(A)$ is the count of times action A has been selected.

Problem 3: Non-Stationary Bandit Problem with Epsilon-Greedy Algorithm

State Space

The state space for the non-stationary bandit problem can also be defined similarly, considering there are 10 actions (one for each bandit):

$$S = \{s_1, s_2, \ldots, s_{10}\}$$

where:

- Each state s_i corresponds to the selection of action a_i.

Action Space

The action space consists of 10 actions:

$$A = \{a_1, a_2, \ldots, a_{10}\}$$

Non-Stationary Epsilon-Greedy Algorithm

The non-stationary epsilon-greedy algorithm adapts to the changing reward distributions. The main differences are:

- After each action selection, the rewards are updated through an independent random walk:

$$R(a) \leftarrow R(a) + \text{random noise}$$

Problem 3 (Updated): Non-Stationary Bandit Problem with Epsilon-Greedy and Learning Rate

Learning Rate (α)

The new addition in this version of the code is the introduction of a learning rate α, which is used to control how much the Q-value is updated at each step. The learning rate is set to:

$$\alpha = 0.7$$

Modified Q-Value Update Rule

Instead of using the standard incremental update rule based on the number of times an action has been selected, the Q-values are now updated using a constant learning rate:

$$Q(A) = Q(A) + \alpha \cdot (R - Q(A))$$

This allows the agent to update the Q-values more quickly, giving more weight to recent rewards, which is especially useful in a non-stationary environment where the reward distribution may change over time.

Bandit Environment

The bandit environment remains non-stationary, with the reward probabilities changing at each step. The reward update function `bandit_nonstat` is unchanged from the previous version, except now the Q-values update with the learning rate α.

Performance Evaluation

The running average of the rewards is calculated as before, and the plot visualizes the average reward over 10,000 steps:

- The plot illustrates the performance of the agent using epsilon-greedy action selection and learning rate-based Q-value updates.
- The average reward over time is depicted using the following formula for updating the running average:

$$R(i) = \frac{(i-1) \cdot R(i-1) + RR}{i}$$

Results

MENACE (Tic-Tac-Toe)

MENACE successfully learns to play Tic-Tac-Toe by reinforcing successful moves and punishing losing ones. Over time, its strategy improves, eventually leading to better performance against random or human opponents.

Binary Bandit Problem (Stationary)

The epsilon-greedy algorithm applied to the binary bandit problem converges towards selecting the action with the higher expected reward. The average reward stabilizes as the agent learns to exploit the better option while occasionally exploring the suboptimal action.

Non-Stationary Bandit Problem

In the non-stationary bandit problem, the modified epsilon-greedy algorithm with a learning rate adapts effectively to changing reward distributions. The Q-values are updated more quickly, allowing the agent to track the best action even as the reward probabilities shift over time. The average reward fluctuates but gradually improves as the agent balances exploration and exploitation.

Repository Link

The github repository link can be found here.

References

[1] Michie, D. (1968). *The Games Machines.* London: J. Murray.
[2] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* Cambridge: MIT Press.