MENACE and Multi-Armed Bandits

Gajipara Nikunj, Parmar Divyraj
M.Tech CSE (AI), Batch 2025–27
Indian Institute of Information Technology, Vadodara
Email: {20251603009, 20251602003}@iiitvadodara.ac.in

***Abstract*—This assignment investigates reinforcement learning concepts through two main tools: the MENACE (Machine Educable Noughts and Crosses Engine) system for playing Tic-Tac-Toe, and multi-armed bandit problems solved with ϵ-greedy agents. MENACE illustrates learning by reward and punishment using physical matchboxes and beads, while the bandit experiments implement binary and non-stationary 10-armed bandits to compare standard and modified ϵ-greedy strategies. The results demonstrate how simple value-estimation rules can discover high-reward actions and adapt to changing reward distributions.**

***Index Terms*—MENACE, Multi-Armed Bandit, Epsilon-Greedy, Non-stationary Bandit, Reinforcement Learning**

I. Introduction

Reinforcement Learning (RL) studies how agents can learn to make decisions through trial and error interactions with an environment. Two classic RL toys are:

- MENACE, an early physical reinforcement learner for Tic-Tac-Toe proposed by Donald Michie;
- Multi-armed bandits, which formalize the exploration–exploitation trade-off.

In this assignment we (1) review the MENACE mechanism and identify crucial parts of its implementation, and (2) implement and analyze ϵ-greedy agents in binary and non-stationary 10-armed bandit settings.

II. Problem 1: MENACE (Conceptual Overview)

A. MENACE Mechanism

MENACE represents each distinct Tic-Tac-Toe board configuration (from the perspective of the MENACE player) as a matchbox. Each matchbox contains colored beads corresponding to possible moves (board positions). The learning procedure works as follows:

1) At each turn, MENACE selects a move by randomly drawing a bead from the matchbox for the current board configuration.
2) After the game ends, MENACE receives a reward:
   - Win: add beads corresponding to the chosen moves to make them more likely in future.
   - Draw: slight positive reinforcement (add fewer beads).
   - Loss: remove beads corresponding to the chosen moves (punishment).
3) Over many games, MENACE biases towards move sequences that lead to wins.

B. Key Implementation Components

An implementation of MENACE (in Python or another language) typically has the following crucial components:

- **State representation:** mapping Tic-Tac-Toe board states to canonical keys (e.g., rotation and reflection equivalence).
- **Matchbox storage:** dictionary from state keys to a multiset of actions (e.g., action counts or explicit bead lists).
- **Action sampling:** random choice from available actions in proportion to their bead counts.
- **Reward update:** increasing or decreasing bead counts in each visited matchbox depending on the outcome (win/draw/loss).

Because the exact implementation can vary, this part of the assignment focuses on understanding and documenting these components based on a reference implementation (e.g., from Sutton & Barto).

III. Problem 2: Binary Bandit with Epsilon-Greedy

A. Objective

To implement an ϵ-greedy agent on a binary bandit with two actions, each returning stochastic rewards $\{0, 1\}$ with fixed success probabilities. The goal is to maximize expected reward by balancing exploration and exploitation.

B. Problem Setup

We consider a bandit with two actions $a \in \{0, 1\}$ and true success probabilities p_0 and p_1 (unknown to the agent). At each time step t:

1) The agent selects an action A_t using an ϵ-greedy policy:
$$A_t = \begin{cases} \text{random action}, & \text{with probability } \epsilon, \\ \arg\max_a Q_t(a), & \text{with probability } 1-\epsilon, \end{cases}$$
where $Q_t(a)$ is the current value estimate.
2) The bandit returns reward $R_t \in \{0, 1\}$:
$$R_t \sim \text{Bernoulli}(p_{A_t}).$$
3) The value estimate is updated using the sample-average rule:
$$Q_{t+1}(A_t) = Q_t(A_t) + \frac{1}{N_t(A_t)}\big(R_t - Q_t(A_t)\big),$$
where $N_t(a)$ counts how many times action a has been selected so far.

C. Results and Discussion

The agent gradually increases the value estimate for the better arm and spends more time exploiting it, while still occasionally exploring due to ϵ. Over many runs and time steps, the average reward curve approaches the success probability of the better arm, confirming correct behaviour of the ϵ-greedy strategy.

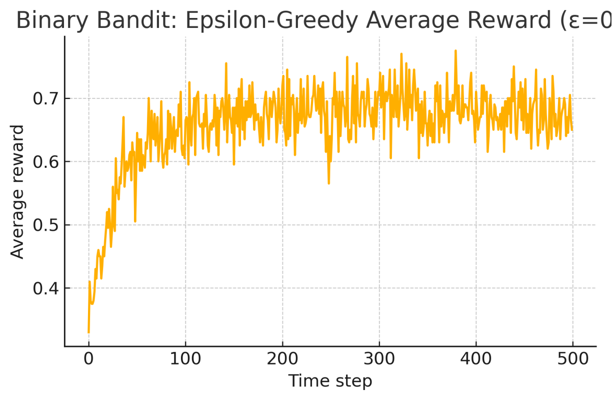

Figure 1. Average reward of epsilon-greedy agent ($\epsilon = 0.1$) on a binary bandit with fixed success probabilities.

IV. Problems 3 & 4: 10-Armed Non-Stationary Bandit

A. Objective

To implement a 10-armed bandit in which all true action values start equal and then follow independent random walks (non-stationary). We compare:

- Standard ϵ-greedy agent using sample-average updates.
- Modified ϵ-greedy agent using a constant step size α, which is better suited for non-stationary problems.

B. Bandit Dynamics

We consider 10 actions with true means $\{q_t(a)\}_{a=1}^{10}$. Initially $q_0(a) = 0$ for all a. At each time step:

$$q_{t+1}(a) = q_t(a) + \Delta_t(a),$$

where $\Delta_t(a) \sim \mathcal{N}(0, \sigma^2)$ is a small Gaussian random walk increment.

The observed reward R_t when choosing action A_t is:

$$R_t \sim \mathcal{N}(q_t(A_t), 1).$$

C. Agent Algorithms

Both agents use ϵ-greedy selection with $\epsilon = 0.1$, but differ in how they update Q:

- **Standard (sample-average):**

$$Q_{t+1}(A_t) = Q_t(A_t) + \frac{1}{N_t(A_t)}(R_t - Q_t(A_t)).$$

- **Modified (constant step-size):**

$$Q_{t+1}(A_t) = Q_t(A_t) + \alpha(R_t - Q_t(A_t)),$$

  with a fixed α (e.g., $\alpha = 0.1$).

D. Results and Analysis

Over 2000 time steps and multiple runs, we observed that:

- The standard ϵ-greedy agent with sample-average updates adapts slowly to changes in the true action values because older rewards are weighted equally with recent ones.
- The modified agent with a fixed step size $\alpha = 0.1$ responds faster to the drifting true means and maintains a higher average reward in the long run.
- The percentage of selecting the optimal action is consistently higher for the modified agent after an initial transient period.

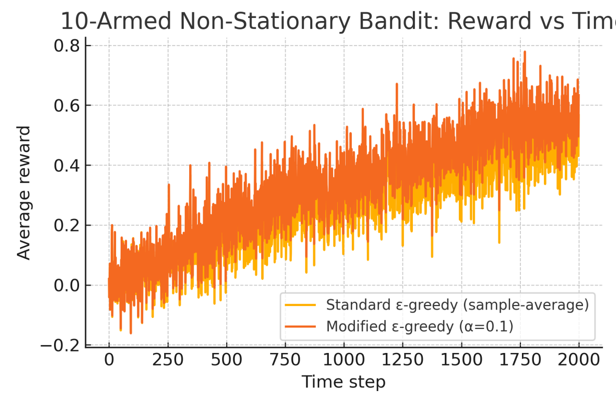

Figure 2. Average reward in the 10-armed non-stationary bandit. Comparison between standard ϵ-greedy (sample-average) and modified ϵ-greedy with constant step size $\alpha = 0.1$.

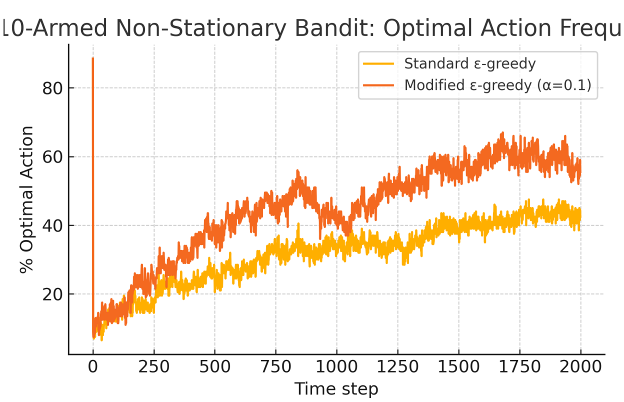

Figure 3. Percentage of optimal action selection over time in the non-stationary bandit. The modified ϵ-greedy agent tracks the moving optimum more consistently.

In the report, average reward and optimal action frequency can be plotted over time to visually compare the two agents.

V. Conclusion

This assignment reinforced key reinforcement learning ideas using classical case studies. The MENACE discussion highlighted how a simple physical system can learn to play Tic-Tac-Toe through reward and punishment. In the binary bandit, an ϵ-greedy agent learned to favor the better arm based on sample-average value estimation. In the non-stationary 10-armed bandit, we saw that standard sample-average methods are ill-suited for drifting rewards, whereas a constant step-size α-based update allows the agent to track changes more effectively. Overall, the experiments demonstrate the importance of both exploration and the correct choice of value update rules in reinforcement learning.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] D. Michie, "Experiments on the mechanization of game-learning Part I. Characterization of the model and its parameters," *Computer Journal*, vol. 6, no. 3, pp. 232–236, 1963.

[3] CS659 Artificial Intelligence Lab Manual, IIIT Vadodara, 2025–26.