Artificial Intelligence Lab Report 7 Reinforcement Learning: Multi-Armed Bandits and MENACE

Anuj Saha, Gadige Nikhil, Divyanshu Ghosh
Indian Institute of Information Technology Vadodara
{202351010, 202351037, 202351036}@iiitvadodara.ac.in

***Abstract*—This report presents the implementation and evaluation of reinforcement learning algorithms for multi-armed bandit problems and the MENACE system. We implemented epsilon-greedy algorithms with both sample-average and constant step-size update rules for stationary and non-stationary bandit environments. Binary bandit experiments demonstrate successful convergence to optimal actions in stationary settings, while the 10-armed non-stationary bandit reveals the superiority of constant step-size ($\alpha = 0.1$) over sample-average methods, achieving 43.67% optimal action selection versus 32.19%. Additionally, we implemented MENACE for tic-tac-toe, successfully training 304 unique game states and achieving progressive improvement from initial 25% to 58% win rates after 50,000 games against a random opponent. Complete code implementation can be found at https://github.com/AnujSaha0111/CS307-Lab-Submissions/tree/main/Submission_7 .**

.

***Index Terms*—Reinforcement Learning, Multi-Armed Bandits, Epsilon-Greedy, MENACE, Non-Stationary Environments, Exploration-Exploitation.**

I. Introduction

Reinforcement learning addresses the fundamental challenge of learning optimal behavior through trial-and-error interaction with an environment [1]. The multi-armed bandit problem serves as a foundational framework for understanding the exploration-exploitation trade-off [8].

In this laboratory, we implemented and evaluated three reinforcement learning approaches:

- **Stationary Binary Bandits**: Two-armed bandits with fixed reward probabilities
- **Non-Stationary 10-Armed Bandits**: Dynamic environments where action values undergo random walks
- **MENACE System**: Michie's 1961 matchbox-based system [2] for tic-tac-toe

We implemented the epsilon-greedy algorithm [1] and compared two value update methods: sample-average ($\alpha = 1/N$) and constant step-size ($\alpha = 0.1$).

II. Problem Statement

A. Binary Bandit Problems (Stationary)

We implemented epsilon-greedy agents for two binary bandit problems:

- **Binary Bandit A**: Probabilities [0.1, 0.2] - low-reward, subtle difference
- **Binary Bandit B**: Probabilities [0.8, 0.9] - high-reward, similar relative difference
- Sample-average updates ($\alpha = 1/N(a)$), $\epsilon = 0.1$, 5,000 steps

B. Non-Stationary 10-Armed Bandit

We developed a 10-armed bandit where the reward landscape continuously changes:

- All means start equal ($q_*(a) = 0$)
- Random walks: $q_(a) \leftarrow q_(a) + \mathcal{N}(0, 0.01)$
- Stochastic rewards: $R \sim \mathcal{N}(q_*(a), 1)$

We compared Standard ($\alpha = 1/N$) vs Modified ($\alpha = 0.1$) agents over 10,000 steps across 200 runs.

C. MENACE Implementation

We implemented MENACE with:

- Canonical state generation for Player 1 (304 states from $3^9 = 19,683$)
- Initial bead counts [8, 4, 2, 1] by ply
- Bead updates: Win (+3), Draw (+1), Loss (-1)
- Training: 50,000 games against random opponent

III. Methodology

A. Epsilon-Greedy Algorithm

We implemented epsilon-greedy for probabilistic action selection.

Algorithm 1 Epsilon-Greedy Action Selection

Require: $Q(a)$, ϵ
Generate $r \sim U(0, 1)$
if $r < \epsilon$ **then**
  $a \leftarrow$ random action
else
  $a \leftarrow \arg\max Q(a)$
end if
return a =0

With probability ϵ, we explore randomly; otherwise, we exploit by selecting the highest-valued action. This simple probabilistic choice effectively balances exploration and exploitation.

B. Value Update Methods

Sample-Average Method: Computes exact mean of observed rewards [1].

$$N(a) \leftarrow N(a) + 1 \tag{1}$$

$$\alpha \leftarrow 1/N(a) \tag{2}$$

$$Q(a) \leftarrow Q(a) + \alpha[R - Q(a)] \tag{3}$$

This method gives equal weight to all past rewards and converges to true action values in stationary environments.

Constant Step-Size Method: Uses fixed α, implementing exponential recency-weighted averaging [1].

$$\alpha \leftarrow 0.1 \tag{4}$$

$$Q(a) \leftarrow Q(a) + \alpha[R - Q(a)] \tag{5}$$

$$Q_n = (1-\alpha)^n Q_0 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \tag{6}$$

The exponential decay favors recent rewards, making it suitable for non-stationary environments.

C. Binary Bandit Environment

Algorithm 2 Binary Bandit Simulation

Require: action $\in \{1, 2\}$, case $\in \{A, B\}$
1: $p \leftarrow [0.1, 0.2]$ if A else $[0.8, 0.9]$
2: Generate $r \sim U(0, 1)$
3: **return** 1 if $r < p$[action $- 1$] else 0 =0

We implemented each bandit as a stochastic process where actions have fixed success probabilities, providing noisy feedback that requires repeated interactions to learn true values.

D. Non-Stationary Bandit Environment

Algorithm 3 Non-Stationary 10-Armed Bandit

Require: action $\in \{1, \ldots, 10\}$
1: Initialize: $q_*(a) \leftarrow 0$ for all a
2: $q_(a) \leftarrow q_(a) + \mathcal{N}(0, 0.01)$ for all a
3: reward $\leftarrow \mathcal{N}(q_*(\text{action}), 1)$
4: **return** reward, $\arg\max q_*(a)$ =0

We introduced non-stationarity through random walks of action values. At each step, all values drift slightly (Gaussian noise, $\sigma = 0.01$), making the optimal action a moving target that models real-world scenarios like evolving user preferences.

E. MENACE System

State Space Generation: We used 8 rotational transformations [2] to identify canonical forms, recursively explored the game tree, and stored only Player 1 states, producing 304 unique positions from $3^9 = 19,683$ total states (64.7:1 compression).

Our MENACE implementation adjusts bead counts after each game: add beads for wins, add few for draws, remove for losses. The $\max(0, \cdot)$ prevents negative beads. Initial counts: Move 1 (8), Move 2 (4), Move 3 (2), Moves 4-5 (1).

Algorithm 4 MENACE Learning Update

Require: game_result, move_history
1: incentive $\leftarrow$ +3 (Win), +1 (Draw), -1 (Loss)
2: **for** each (state, move) in move_history **do**
3: beads[state][move] $\leftarrow$
4: $\max(0, \text{beads[state][move]} + \text{incentive})$
5: **end for**=0

F. Experimental Design

Binary Bandits: We used epsilon-greedy with $\epsilon = 0.1$ and sample-average updates for 5,000 steps per trial. We measured final Q-values, action counts, and average reward to evaluate convergence quality.

Non-Stationary Bandits: We compared Standard ($\alpha = 1/N$) versus Modified ($\alpha = 0.1$) agents over 10,000 steps each, repeating this for 200 independent runs to ensure statistical reliability. Our primary metric was the percentage of optimal action selections.

MENACE: We trained our implementation for 50,000 games against a random opponent, dividing the training into 10 evaluation blocks of 5,000 games each. We tracked Win/Draw/Loss rates per block to observe learning progression over time.

IV. Results

A. Binary Bandit Performance

TABLE I
BINARY BANDIT RESULTS (5,000 STEPS, $\epsilon = 0.1$)

Bandit A	**Action 1**	**Action 2**	**Metrics**
True Probability	0.1	0.2	Optimal: Act 2
Final Q-Value	0.0931	0.2043	Error: 2.15%
Action Count	541	4459	Exploit: 89.2%
Average Reward: 0.1936			
Bandit B	**Action 1**	**Action 2**	**Metrics**
True Probability	0.8	0.9	Optimal: Act 2
Final Q-Value	0.7891	0.9015	Error: 0.17%
Action Count	539	4461	Exploit: 89.2%
Average Reward: 0.8946			

Our agents successfully identified Action 2 as optimal in both cases. The Q-values closely matched true probabilities - within 2.15% for Bandit A and just 0.17% for Bandit B. Action counts showed consistent 89.2% exploitation, perfectly aligning with our $\epsilon = 0.1$ setting that reserves 10% for exploration. This consistency across both low-reward and high-reward scenarios validates that our implementation is working correctly.

B. Non-Stationary Bandit Performance

We observed a striking difference between the two approaches. Our Modified agent significantly outperformed the Standard agent, selecting the optimal action 43.67% of the time compared to just 32.19%. This 35.7% relative improvement is highly statistically significant ($p < 0.001$), confirming this isn't due to random chance.

TABLE II
NON-STATIONARY PERFORMANCE (200 RUNS, 10,000 STEPS)

Agent	Step-Size α	Optimal %	Std Dev
Standard	$1/N(a)$	32.19%	4.73%
Modified	0.1 (constant)	43.67%	5.12%
Improvement: 35.7% — t-stat: 16.52 — $p < 0.001$			

The reason becomes clear when we consider how the agents adapt to change. The Standard agent's decreasing step-size ($\alpha = 1/N$) becomes negligible after many selections - dropping to just 0.001 after 1000 trials, meaning new observations barely influence its beliefs. In contrast, our Modified agent maintains $\alpha = 0.1$ throughout, allowing continuous adaptation to the changing reward landscape. The effective memory window of $1/\alpha = 10$ steps proved well-suited for tracking the random walk with $\sigma = 0.01$ per step.

C. MENACE Learning Progress

Our MENACE implementation showed remarkable learning progression. Starting from near-random play (25% wins), it steadily improved to 58% wins with strong linearity ($R^2 = 0.982$). We observed three phases: **Exploration** (Blocks 1-3) with high losses (34%→26%) as the system explored strategies; **Strategy Formation** (Blocks 4-7) with accelerated improvement (34%→43% wins) and declining losses (22%→11%); **Optimization** (Blocks 8-10) with continued growth to 58% wins as aggressive winning strategies replaced defensive draws.

V. DISCUSSION

A. Epsilon-Greedy Balance

Our choice of $\epsilon = 0.1$ provided effective exploration-exploitation balance [1], successfully discovering optimal actions in stationary bandits and tracking non-stationary shifts. The 10% exploration rate proved sufficient to prevent getting stuck in suboptimal policies while dedicating 90% of actions to exploiting learned knowledge. Future work could investigate decaying schedules [6] or adaptive ϵ based on uncertainty estimates.

B. Step-Size Comparison

Constant $\alpha = 0.1$ outperformed sample-average by 35.7% in our non-stationary experiments. The fundamental issue with sample-average is that $\alpha = 1/N(a)$ approaches zero as actions are selected repeatedly. After 1000 selections of an action, the step-size drops to 0.001, meaning new observations have almost no impact on the value estimate [1]. The agent essentially becomes increasingly confident in its outdated beliefs.

Our constant step-size maintains its learning rate throughout, creating exponential weighting where recent rewards have approximately 10× more influence than those from 23 steps ago. This effective "forgetting" of old information is exactly what we need when the world is changing. The memory window of $1/\alpha = 10$ steps matched our problem well. With the random walk adding $\mathcal{N}(0, 0.01)$ noise at each step, we expected action values to diverge by about $\mathbb{E}[|\Delta q_*|] = 0.01\sqrt{10000} = 1.0$ over 10,000 steps - comparable to the reward noise ($\sigma = 1$) [5]. Despite this challenge, our Modified agent tracked the optimal action reasonably effectively.

C. MENACE Learning Characteristics

Reinforcement Schedule: The asymmetric rewards (+3/+1/-1) proved highly effective [2]. Consider the cumulative impact: a single win adds 15 beads total (5 moves × 3 beads each) while a single loss removes only 5 beads. This 3:1 ratio means winning strategies are reinforced much more strongly than losing strategies are discouraged, preventing catastrophic forgetting when the system encounters streaks of bad luck. Even after several losses, a few wins can quickly restore bead counts and recover good strategies.

State Space Efficiency: By exploiting rotational symmetry [2], we compressed the full state space of $3^9 = 19,683$ board configurations down to just 304 unique positions - a remarkable 64.7:1 compression ratio. This reduction made physical matchbox implementation feasible in 1961 and demonstrates how clever problem formulation can be as important as sophisticated algorithms.

Learning Phases: We observed genuine strategic development across three distinct phases. Early exploration (Blocks 1-3) was necessary but painful, with high losses as the system tried different strategies. The middle phase (Blocks 4-7) showed consolidation, with the system learning to avoid obvious mistakes. The late phase (Blocks 8-10) demonstrated sophistication, actively pursuing wins rather than settling for safe draws. The strong linear trend ($R^2 = 0.982$) throughout all 50,000 games suggests learning capacity wasn't fully saturated, indicating potential for further improvement with additional training.

D. Practical Applications

The techniques we explored have broad real-world applicability that extends beyond academic exercises.

Constant Step-Size Learning:

Online recommendation systems where user preferences evolve over time, Adaptive control systems in manufacturing or robotics where operating conditions change and Financial portfolio optimization dealing with non-stationary market returns.

Epsilon-Greedy Exploration:

- A/B testing in web development with explicit exploration constraints.
- Clinical trials requiring ethical balance between exploring treatments and exploiting known effective ones.
- Resource allocation problems under uncertainty in operations research.

MENACE-Style Learning:

- Game AI for simple board games where interpretability matters.
- Educational demonstrations of machine learning concepts for non-technical audiences.
- Explainable AI systems where stakeholders need to understand policy decisions.

TABLE III
MENACE TRAINING PERFORMANCE (50,000 GAMES)

Block	Games	Wins	Draws	Losses	Win %	Draw %	Loss %
1	5,000	1,247	2,031	1,722	24.94	40.62	34.44
2	10,000	1,401	2,076	1,523	28.02	41.52	30.46
3	15,000	1,533	2,143	1,324	30.66	42.86	26.48
4	20,000	1,689	2,201	1,110	33.78	44.02	22.20
5	25,000	1,812	2,267	921	36.24	45.34	18.42
6	30,000	1,978	2,289	733	39.56	45.78	14.66
7	35,000	2,156	2,314	530	43.12	46.28	10.60
8	40,000	2,389	2,203	408	47.78	44.06	8.16
9	45,000	2,634	2,012	354	52.68	40.24	7.08
10	50,000	2,891	1,856	253	57.82	37.12	5.06
Linear Regression: Win% $= 18.67 + 3.91\times$ Block — $R^2 = 0.982$ — $p < 0.001$							

VI. CONCLUSION

In this lab, we successfully implemented and evaluated fundamental reinforcement learning algorithms across three problem domains, each revealing different aspects of the learning challenge. Our key findings:

- **Stationary convergence**: Sample-average methods worked excellently when reward probabilities stayed fixed. We achieved Q-values within 2% of true values after 5,000 steps [1], confirming theoretical predictions that sample averages converge to true means given sufficient data.
- **Non-stationary superiority**: Constant step-size ($\alpha = 0.1$) delivered 35.7% improvement over sample-average (43.67% vs 32.19% optimal actions) when action values drifted [1]. This highlights a crucial lesson: the "best" algorithm depends critically on whether the environment is stationary. This has direct implications for real-world systems where preferences, conditions, or parameters evolve over time.
- **MENACE success**: Watching MENACE learn was particularly satisfying. Starting from 25% wins, it improved to 58% with remarkable consistency ($R^2 = 0.982$). The simplicity of bead-counting [2] proves that effective learning doesn't always require complex mathematics - sometimes elegant simplicity is most powerful.
- **Exploration balance**: Our $\epsilon = 0.1$ choice worked well across all experiments [1], though in hindsight we might have tested decaying rates to reduce exploration as learning progressed.

We discovered fundamental trade-offs in reinforcement learning: sample-average is optimal for stationary environments where past observations remain relevant, but constant step-size excels when adaptation to change is necessary [5]. Neither approach is universally superior - the context determines the winner. Our results empirically validated exponential recency weighting theory [1] for non-stationary settings.

MENACE deserves special recognition for demonstrating that effective learning can emerge from remarkably simple mechanisms. The bead-counting approach, combined with symmetry exploitation to manage state space complexity [2], achieves impressive results while remaining completely interpretable - a valuable property in modern AI applications where explainability is increasingly important.

Looking ahead, several promising extensions merit investigation: UCB action selection for more principled exploration [3], gradient bandits with softmax action preferences [1], contextual bandits that incorporate state information [8], training MENACE through self-play against increasingly competent opponents, and developing theoretical regret bounds for constant step-size methods in non-stationary settings [3], [6]. Each direction could deepen our understanding of when and why different learning strategies succeed.

REFERENCES

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

[2] D. Michie, "Experiments on the mechanization of game-learning part I," *The Computer Journal*, vol. 6, no. 3, pp. 232–236, 1963.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[4] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2011.

[5] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.

[6] E. Even-Dar, S. Mannor, and Y. Mansour, "Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems," *Journal of Machine Learning Research*, vol. 7, pp. 1079–1105, 2006.

[7] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, p. 103535, 2021.

[8] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, UK: Cambridge University Press, 2020.