LLM Usage Declaration

Name: Om Mallick

Roll No: 251110055

Email: ommallick25@iitk.ac.in

==========

It is ok to use LLM for understanding concepts, ideation, and initial code stubs but not for completely solving the assignment; even if you use LLM for the above mentioned things, you should be able to explain the solution without the LLM during the viva (if you are called for the viva).

==========

Q1. Which Large Language Model (LLM) did you use to complete this assignment?

Kimi K2.5, developed by Moonshot AI

Q2. In which questions did you use the LLM, and how did it help you? (you can extend this table and add more rows): In case the answer to above question is NO then you can write "NA" and no need to make the table.

Question Number/ Sub Parts	**Used LLM? (Yes/ No)**	**Purpose of Using LLM (e.g., idea generation, explanation, code help, grammar correction)**	**Brief Description of LLM Contribution**
Q1.1 (2-armed Bernoulli	Yes	Code structure and debugging.	Provided the Gymnasium environment template with reset() and step() methods. I modified the reward logic to

Bandit environment)			match Bernoulli distribution and added validation checks.
Q1.2 (10-armed Gaussian Bandit environment)	Yes	Explanation and code help	Explained how to sample q*(k) from N(0,1) and rewards from N(q*(k), sigma^2). Provided structure for the Gaussian bandit; I implemented the reset logic to resample q_true values.
Q1.3(a) Greedy Agent	Yes	Debugging critical error	My initial implementation got stuck on arm 0 due to np.argmax() tie-breaking. LLM explained the optimistic initialization concept. I implemented the fix using np.ones() * 2.0 and verified it worked.
Q1.3(b-f) Other Agents	Partially	Algorithm pseudocode verification	I implemented epsilon-greedy, decaying epsilon-greedy, Softmax, and UCB based on lecture slides. Used LLM to verify my UCB exploration bonus formula was correct.
Q1.4-Q1.5 (Testbed experiments)	No	NA	Implemented run_experiment functions and plotting independently based on lecture examples and numpy documentation.

Q1.6-Q1.7 (Regret calculations)	No	NA	Derived regret formulas from lecture definitions and implemented independently.
Q1.8-Q1.11 (Optimal action %, log scale plots)	No	NA	Extended existing plotting functions, no LLM assistance.
Q2: Random Walk Environment	Yes	Bellman equation setup	Verified my system of linear equations for true value calculation was set up correctly. I solved the matrix equation independently.
Q2: MC and TD Prediction	Partially	Algorithm structure	Provided general loop structure for episode generation. I implemented FVMC vs EVMC distinction and TD bootstrapping logic based on lecture equations.
Q2: Target collection functions	Yes	Debugging	Helped debug why my MC return calculation was incorrect - needed to process trajectory backwards. I fixed the implementation.

Q3. Did you refer to any other sources or websites apart from the LLM and lecture slides?

Source	**Contribution**

Gymnasium Documentation (gymnasium.farama.org)	Referenced for spaces.Discrete, Env class structure, and proper reset()/step() return signatures. Used to verify environment API compliance.
Sutton & Barto, "Reinforcement Learning: An Introduction" (2nd Ed.)	Chapter 2 (Multi-armed Bandits) for UCB formula derivation and exploration/exploitation trade-offs. Chapter 5 for Monte Carlo and TD algorithm pseudocode.
NumPy Documentation (numpy.org)	Referenced for np.random.choice() with probabilities (Softmax), np.argmax() behavior, and efficient array operations for the testbed experiments.
Matplotlib Documentation	Referenced for semilogx(), axhline(), and legend handling in comparative plots.

Q4. Apart from using any LLM, what was **your own contribution** while solving this assignment?

Conceptual Understanding

- Derived the Bellman equations for the Random Walk Environment true values by setting up the system of linear equations for non-terminal states, with boundary conditions V(0)=0, V(6)=0.
- Understood why FVMC vs EVMC differ in variance characteristics by analyzing the correlation structure of multiple visits within an episode.
- Analyzed the exploration-exploitation tradeoff by designing and interpreting comparative experiments across different epsilon values, UCB c parameters, and temperature schedules.

Implementation Effort

- Designed the experimental framework: Created run_experiment_bernoulli() and run_experiment_gaussian() functions to average over 50-200 environment instances, ensuring statistically significant results.

- Handled edge cases: Added proper handling for non-terminating trajectories (returning empty lists), division-by-zero protection in UCB, and seed management for reproducibility.
- Extended basic algorithms: Implemented decay schedules (linear/exponential) for epsilon-greedy and Softmax, and created multiple visualization functions for different analysis types (reward curves, regret, optimal action %).

Experimental Analysis

- Ran all experiments multiple times with different random seeds to verify consistency of results.
- Tuned hyperparameters manually: Determined that epsilon=0.1 works better than epsilon=0.5 for epsilon-greedy, and c=0.5 is effective for UCB through systematic experimentation.
- Interpreted all plots: Identified that TD converges faster than MC due to bootstrapping, that greedy gets stuck without optimistic initialization, and that UCB achieves logarithmic regret—all observations written independently.

Debugging and Verification

- Traced through algorithm execution manually for short episodes to verify MC return calculations and TD updates were correct.
- Compared empirical results to theoretical predictions: Verified that Bernoulli bandit empirical means matched aplha, beta parameters and that Gaussian bandit sample standard deviations matched sigma.

Code Organization

- Structured all code into reusable functions with proper docstrings.
- Ensured consistent random seeding across environment resets and agent runs for fair comparison.
- Created modular plotting functions that could handle both linear and log scales, single and averaged runs.