RL Assignment 1 Programming Question

Question: Modified Gradient Bandit with Adaptive Baseline

Problem Statement

Implement a **Gradient Bandit Algorithm** with an **adaptive baseline** that automatically adjusts based on the variance of recent rewards. Your implementation should include the following components:

Background: The standard gradient bandit algorithm uses a baseline (typically the average reward) to reduce variance in updates. However, when reward distributions have different variances across arms or change over time, a simple average may not be optimal.

Task

Implement a gradient bandit agent with the following specifications:

1. **Standard Gradient Bandit Update:**
    a. For selected action: $H[a] = H[a] + \alpha(R - baseline)(1 - \pi[a])$
    b. For non-selected actions: $H[a] = H[a] - \alpha(R - baseline)\pi[a]$
    c. Action probabilities: $\pi[a] = \exp(H[a]) / \Sigma \exp(H[b])$
2. **Adaptive Baseline:** Instead of using just the average reward, use a **weighted baseline** that adapts based on recent reward variance:

$$baseline\_t = (1 - \beta) * avg\_reward + \beta * recent\_variance\_adjusted\_mean$$

where:

- avg_reward is the running average of all rewards
- $recent\_variance\_adjusted\_mean$ considers the last W rewards with weights inversely proportional to their squared deviation from the mean
- β is an adaptation parameter (0 ≤ β ≤ 1)

3. **Performance Tracking:**
    a. Track the percentage of optimal actions selected
    b. Track cumulative regret (difference from optimal arm)
    c. Compare against standard gradient bandit (β = 0)

Requirements

1. Implement the *AdaptiveGradientBandit* class with methods:
    a. *__init__(n_arms, alpha, beta, window_size)*
    b. *select_action()* - returns action based on softmax policy
    c. *update(action, reward)* - updates preferences using adaptive baseline
    d. *get_baseline()* - returns current baseline value
2. Run experiments on a **non-stationary 10-armed bandit** where:
    a. True reward means start at random values from N(0, 1)
    b. Every 500 steps, add N(0, 0.1) to each arm's true mean (random walk)
    c. Actual rewards are sampled from N(true_mean, 1)
3. Compare performance over 2000 steps for:
    a. Standard gradient bandit (β = 0)
    b. Adaptive gradient bandit (β = 0.3)
    c. Adaptive gradient bandit (β = 0.6)
4. Plot:
    a. Average reward over time (running average over last 100 steps)
    b. Percentage of optimal action selection
    c. Baseline values over time for different β values

Function Signatures:

// Add function signatures here.

Expected Observations

- Adaptive baselines should perform better in non-stationary environments
- Higher β values should show faster adaptation but potentially more variance
- Baseline values should track the changing reward landscape