fast-logo

**CS3018 - Reinforcement Learning**

**Assignment 1**

**Due Date: 27th February, 2025**

You are given a k-armed bandit problem, where an agent selects among k slot machines (arms), each with an unknown reward distribution. Your task is to implement and compare different action selection strategies in a simulated environment.

### **1. Implement a k-Armed Bandit Simulator**

#### **Task:**

* Write a Python class KArmedBandit(k), where:
  + **Each arm**’s true value is randomly initialized at the start from a normal distribution with mean 0 and variance 1 i.e.
  + Calling pull\_arm(a) should return a **random reward** sampled from normal distribution .

**Expected Outcome:**

bandit = KArmedBandit(10)

print(bandit.pull\_arm(2)) # Should return a random reward from a normal distribution

**2. Implement ε-Greedy Action Selection**

**Task:**

* Implement an agent that chooses the best-known arm with probability 1−ε and a random arm with probability ε
* Maintain an estimate of the average reward for each action.
* Update the action-value estimates using the incremental mean formula:

**Expected Outcome:**

agent = EpsilonGreedyAgent(10, 0.1)

action = agent.select\_action()

reward = bandit.pull\_arm(action)

agent.update(action, reward)

print(agent.q\_values)

**3. Implement Optimistic Initial Values**

**Task:**

Modify the ε-greedy agent to start with high initial estimates for Q(a) instead of zero. This will encourage exploration early on.

**Instructions:**

Modify the EpsilonGreedyAgent class to initialize q\_values to a high value (e.g., 5.0).

Run experiments comparing the optimistic initialization to regular ε-greedy.

**Expected Outcome:**

Optimistic initialization encourages the agent to explore more in the beginning.

**4. Implement Upper Confidence Bound (UCB) Algorithm**

**Task:**

Implement UCB agent that selects action using the following formula:

where:

Q(a) is the estimated value of action .

N(a) is the number of times action a was chosen.

t is the total number of steps taken.

c is a confidence level (e.g., 2.0).

**Expected Outcome:**

UCB will select arms optimally, balancing exploration & exploitation.

agent = UCB\_Agent(10, 2)

action = agent.select\_action()

reward = bandit.pull\_arm(action)

agent.update(action, reward)

print(agent.q\_values)

**5. Experiment: Compare Strategies:**

**Task:**

Run simulations comparing:

* ε-greedy (ε = 0.1)
* Optimistic ε-greedy (Q-values initialized high)
* UCB

Track average reward over time and percentage of optimal actions selected.

**6. Visualization Task**

**Task:**

* Plot the average reward over time for ε-greedy, optimistic, and UCB.
* Plot the optimal action selection percentage over time.
* Compare results and analyze which strategy performs best.

**Expected Output:**

* UCB should converge faster to the optimal arm.
* Optimistic initialization helps in early exploration.
* ε-greedy performs decently but requires fine-tuning of ε.