K-armed bandits

The **K-armed bandit problem** is a classic problem in reinforcement learning and decision theory, often used to study **exploration vs. exploitation trade-offs**. The problem is named after a slot machine (or "one-armed bandit") with multiple arms (K arms), where each arm provides a different unknown probability distribution of rewards. The goal is to maximize the total reward over a series of trials by deciding which arms to pull based on past outcomes.

Problem Setup

- There are K independent arms, and each arm iii provides a reward Ri, drawn from an unknown probability distribution Pi.
- The agent interacts with the environment by pulling one arm at each time step, and the objective is to maximize the cumulative reward over time.

Objectives

The main goal is to identify which arm has the highest expected reward and pull that arm as frequently as possible to maximize the overall gain. However, since the rewards are unknown at the start, the agent must decide whether to explore (try new arms to gather more information) or exploit (stick with the best-performing arm so far).

Key Concepts

1. Exploration vs. Exploitation Trade-off

- **Exploration**: Trying out different arms to gather information about their reward distributions.
- **Exploitation**: Pulling the arm that has provided the highest reward in the past to maximize the immediate payoff.
- The challenge is to balance these two strategies: if you explore too much, you may miss out on high rewards from known arms, but if you exploit too early, you may miss discovering better arms.

2. Action-Value Estimate

Let Qt(a) be the estimated value (expected reward) of pulling arm a at time t. Initially, all Q0(a) values are set to zero, and the agent updates them after receiving each reward.

The **sample-mean method** to estimate the value of an arm is:

$$Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{N_t(a)} R_i$$

Where:

- $N_t(a)$ is the number of times arm a has been pulled by time t.
- Ri is the reward received after pulling arm a.

3. Regret

Regret measures how much worse the cumulative reward is compared to always pulling the optimal arm. Formally, the regret at time T is:

$$\text{Regret}(T) = T\mu^* - \sum_{t=1}^{T} r_t$$

Where:

- μ∗ is the expected reward of the optimal arm.
- r_t is the reward at time t.

Algorithms for Solving the K-armed Bandit Problem

There are several strategies to solve the K-armed bandit problem:

1. Greedy Algorithm

The **greedy algorithm** always chooses the arm with the highest estimated reward, based on past experience:

$$a_t = \arg\max_a Q_t(a)$$

However, this purely exploits the known information and does not explore new possibilities, potentially leading to suboptimal performance if a better arm has not been sufficiently explored.

2. ϵ\epsilonϵ-Greedy Algorithm

The **ϵ\epsilonϵ-greedy algorithm** balances exploration and exploitation by choosing:

- A random arm with probability ϵ\epsilonϵ (exploration),
- The arm with the highest estimated reward with probability 1−ϵ.

This way, the agent explores enough to avoid getting stuck with suboptimal arms while still exploiting the best-known arm most of the time.

3. UCB (Upper Confidence Bound)

The **UCB algorithm** adds an exploration bonus to the action-value estimates. It pulls the arm that maximizes:

$$a_t = \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right)$$

Where ccc controls the degree of exploration, and the second term encourages the selection of arms that have been played fewer times (hence less certain).

4. Thompson Sampling

In **Thompson Sampling**, each arm is assigned a probability distribution (posterior distribution) over its expected reward. The agent pulls an arm according to the probability that it is the best arm based on these posterior distributions.

Example: 3-Armed Bandit Problem

Suppose you have 3 slot machines (arms), each with different reward probabilities:

- Arm 1: P1∼N(1,1)
- Arm 2: P2∼N(2,1)
- Arm 3: P3∼N(3,1)

Each time you pull an arm, you receive a reward sampled from the corresponding probability distribution. Initially, you don't know which arm is best, and you need to balance exploring the arms and exploiting the best one.

1. Start by pulling each arm once to get an initial estimate.
2. Use the ϵ\epsilonϵ-greedy or UCB algorithm to decide which arm to pull next.
3. After each pull, update your estimate of the expected reward for that arm.

Conclusion

The K-armed bandit problem provides a fundamental model for studying how to balance exploration and exploitation in decision-making. Different algorithms such as ϵ\epsilonϵ-greedy, UCB, and Thompson Sampling help in addressing this challenge in various settings, from online advertising to clinical trials and