Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 12: Multi-Armed Bandits

Summarized By: Behnia Soleymani

- Imagine you're faced with several options (like different slot machines, medical treatments, or website designs), each yielding rewards with an unknown probability or average value. Your goal is to **maximize your total reward** over a series of choices. The catch is, you don't initially know which option is best. You have to figure it out by trying them, learning from the outcomes (rewards) you receive. This is the essence of the Multi-Armed Bandit problem.
  - **Key Elements**
    - **Agent:** The decision-maker.
    - **Arms (Actions):** The set of k options available.
    - **Reward:** A numerical value received after choosing an arm, often drawn from a probability distribution specific to that arm.
    - **True Action Value ($q^*(a)$):** The theoretical average reward of arm a. This is unknown to the agent.
    - **Objective:** Maximize cumulative reward over time by identifying and pulling the best arm(s).
- **The Fundamental Dilemma: Exploration vs. Exploitation** This is the central challenge in MAB and many related learning problems. At each decision point, the agent must choose between:
  - **Exploitation:** Choosing the arm that *currently seems best* based on past experience. This leverages existing knowledge to get good rewards *now*.
  - **Exploration:** Choosing a different arm, possibly one that looks suboptimal or hasn't been tried much. The goal is to gather more information, potentially discovering a truly better arm for *future* rewards, even if it means sacrificing immediate gain.

  Striking the right balance is crucial. Pure exploitation risks getting stuck with a suboptimal choice forever. Pure exploration means constantly trying options without capitalizing on good discoveries.
- **Estimating Arm Values** Since the true values ($q^*(a)$) are unknown, the agent must estimate them based on observed rewards. Let $Q(a)$ denote the estimate for arm a.
  - **Sample Average** A common method is to estimate the value of arm a as the average of all rewards received from that arm so far:

$$Q(a) = \frac{\text{Sum of rewards received from arm } a}{\text{Number of times arm } a \text{ was chosen}}$$

  - **Incremental Updates** To avoid recalculating the sum and count each time, we can update the estimate efficiently after receiving the n-th reward (R_n) for a specific arm (let Q_n be the estimate after $n-1$ rewards for that arm):

$$Q_{n+1} = Q_n + \text{StepSize} \times (R_n - Q_n)$$

    For the simple sample average, the StepSize after receiving the n-th reward is $1/n$.

- **Handling Change: Non-Stationary Problems** What if the true reward probabilities ($q^*(a)$) change over time? (e.g., a website layout's effectiveness fades).
  - **Problem with Sample Average:** The $1/n$ step size eventually becomes very small, making the estimate resistant to change and slow to adapt to new true values. Old, potentially irrelevant information is weighted equally with new information.
  - **Solution: Constant Step Size:** Use a fixed step size α (e.g., $\alpha = 0.1$) instead of $1/n$:

$$Q_{n+1} = Q_n + \alpha(R_n - Q_n)$$

  - **Benefit:** This gives more weight to recent rewards and effectively "forgets" older rewards exponentially. This allows the estimate to track changes in the underlying true values.
- **Strategies for Balancing Exploration and Exploitation** Several algorithms implement different ways to manage this trade-off:
  - **Epsilon-Greedy (ϵ-Greedy)**
    - **How it works:** With probability $1 - \epsilon$, exploit (pick the arm with the highest current $Q(a)$). With probability ϵ, explore (pick a random arm uniformly).
    - **Pros:** Very simple to understand and implement.
    - **Cons:** Exploration is "dumb" – it doesn't prioritize exploring more promising or less-understood arms; it might waste time re-exploring known bad arms.
  - **Optimistic Initial Values**
    - **How it works:** Initialize all $Q(a)$ estimates to a value much higher than any possible true reward. Then, always act greedily (choose the arm with the max $Q(a)$).
    - **Mechanism:** Trying an arm will likely result in a reward lower than the initial optimistic value, reducing its $Q(a)$. The agent is thus encouraged to try other arms (which still have the high initial value) until all arms have been tried and their estimates become more realistic.
    - **Pros:** Simple way to encourage initial exploration.
    - **Cons:** Performance can be sensitive to the choice of the initial optimistic value; exploration might cease too early.
  - **Upper Confidence Bound (UCB)**
    - **How it works:** Selects the arm that maximizes a combination of the current estimate and an "uncertainty bonus". Let t be the current time step, and $N_t(a)$ be the number of times arm a has been selected prior to t. Choose arm A_t such that:

$$A_t = \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right]$$

    - ∗ **Components:**
        - $Q_t(a)$: The current estimate (exploitation term).
        - $\sqrt{\frac{\ln(t)}{N_t(a)}}$: The uncertainty bonus (exploration term). It increases with time ($\ln(t)$) and decreases as an arm is pulled more often ($N_t(a)$).
        - c: A parameter controlling the amount of exploration ($c \geq 0$).
    - ∗ **Mechanism:** Explicitly favors arms that either look good ($Q_t(a)$ is high) or haven't been tried much ($N_t(a)$ is low). It directs exploration intelligently towards uncertain options.
    - ∗ **Pros:** Often more efficient exploration than ϵ-greedy, leading to better performance. Principled approach based on confidence bounds.
    - ∗ **Cons:** Slightly more complex calculation. Assumes $N_t(a) > 0$; typically requires playing each arm once initially.

- **Contextual Bandits: Adding Side Information**
    - **Concept:** In many real-world scenarios, the best action depends on the current situation or *context* (e.g., recommending different products based on user profile, time of day, etc.).
    - **Process:** The agent observes context features, then chooses an arm, receives a reward, and learns how context influences rewards for different arms.
    - **LinUCB Algorithm:** A popular method for contextual bandits.
        - ∗ **Key Assumption:** Assumes the expected reward is a *linear function* of the context features $x_{t,a}$ specific to that arm a at time t: $E[r|x_{t,a}] = x_{t,a}^T \theta_a^*$, where θ_a^* is an unknown weight vector for arm a.
        - ∗ **Mechanism:** Uses linear regression (specifically, Ridge Regression) to estimate the weight vector $\hat{\theta}_a$ for each arm. It then applies the UCB principle, selecting arms based on both the predicted reward ($x_{t,a}^T \hat{\theta}_a$) and the uncertainty of that prediction for the current context $x_{t,a}$.

- **Thompson Sampling: A Bayesian Perspective**
    - **Philosophy:** Instead of maintaining a single point estimate $Q(a)$, maintain a full *probability distribution* (a "belief") over what the true value $q^*(a)$ (or underlying parameter, like success probability θ_a) might be.
    - **Mechanism:**
        1. **Sample:** At each step, draw one possible value for each arm's parameter from its current belief distribution.
        2. **Act:** Choose the arm whose *sampled* value is the highest.
        3. **Update:** Observe the reward and use Bayes' theorem to update the belief distribution for the arm that was chosen (making the belief more accurate).

- **Example (Bernoulli Rewards):** If rewards are $0/1$ (failure/success), the unknown parameter is the success probability θ_a. We can use a **Beta distribution** to represent our belief about θ_a. Observing a success or failure allows us to easily update the parameters of the Beta distribution (due to conjugacy). If the belief is Beta(α, β), $\alpha - 1$ can be seen as successes and $\beta - 1$ as failures. Observing a success updates to Beta$(\alpha + 1, \beta)$, a failure updates to Beta$(\alpha, \beta + 1)$.
- **Pros:** Often performs very well empirically, provides a natural way to encode uncertainty and perform exploration.
- **Cons:** Requires specifying prior beliefs, relies on sampling which introduces randomness into the action selection.