Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 19: Exploration Methods

Summarized By: Behnia Soleymani

- **Why is Exploration Hard? Sparse Rewards & Long Time Horizons**
  - **Easy Problems** (e.g., **Breakout**): Rewards are frequent (hitting bricks). Simple strategies often work.
  - **Hard Problems** (e.g., **Montezuma's Revenge**):
    - **Sparse Rewards:** Meaningful rewards (getting a key, opening a door) are rare and only occur after specific, sometimes long, sequences of actions.
    - **Temporally Extended Tasks:** The actions needed to get a reward are separated by many steps where nothing obviously good happens.
    - **Complex Dependencies:** Often requires solving multiple sub-tasks in a specific order (get key $\rightarrow$ go to door $\rightarrow$ use key).
    - **Misleading Feedback:** Bad outcomes (like hitting a skull) might give zero reward, not negative, making it hard to learn they are bad quickly.
  - Humans use understanding (key opens door), but RL agents start from scratch.
- **Simple Exploration Can Fail: The Epsilon-Greedy Limitation**
  - **The Problem in Complex Tasks:** If you need to master k sequential sub-tasks and then explore for the $(k+1)^{th}$:
    - You must *exploit* correctly for roughly k steps, then *explore* correctly at the $(k+1)^{th}$ step.
    - The probability of doing this is roughly $(1-\epsilon)^k\epsilon^1$ (assuming $O(k)$ exploit steps, $O(1)$ explore step). This probability decreases *exponentially* as the sequence length k increases. so epsilon-greedy performs poorly on tasks requiring long, specific action sequences.
    - **Example:** For $\epsilon = 0.1$, $k = 5$, the chance is only $\approx 6\%$. For $\epsilon = 0.5$, $k = 5$, it's $\approx 3\%$. It becomes very unlikely to explore the *right* thing at the *right* time.
- **Measuring Exploration Success: Regret**
  - How good is an exploration strategy? We can measure its **Regret**.
  - **Definition:** Regret compares the total reward the agent *actually* got to the reward it *could have* gotten if it knew the best strategy from the start.
  - **Formula:**

$$\text{Reg}(T) = T \cdot \mathbb{E}[r(a^*)] - \sum_{t=1}^{T} r(a_t)$$

    - T: Total number of time steps (or episodes).
    - $\mathbb{E}[r(a^*)]$: Expected reward of the single best *fixed* action/policy (a^*) in hindsight.
    - $\sum r(a_t)$: The sum of actual rewards received by the agent's chosen actions a_t over time.
  - **Goal:** Design exploration algorithms with low (ideally sub-linear, like $O(\log T)$) regret.

- **Learning from Simplified Problems: Multi-Armed Bandits (MAB)**
  - **Bandits:** A simpler RL setting with no states, just actions ("arms"). Choose an arm, get a reward from an unknown distribution for that arm. Focuses purely on exploration vs. exploitation.
  - Two powerful strategies emerged from bandit research:
    - **A) UCB:**
      - **Mechanism:** Estimate the average reward $\hat{\mu}_a$ for each arm a. Add a bonus based on uncertainty. Choose the arm maximizing this optimistic estimate. choose arm A_t at time t:

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

    - **B) Posterior Sampling (Thompson Sampling):**
      - **Principle:** Act according to the *probability* that an action is the best one, based on current beliefs.
      - **Mechanism:**
        1. Maintain a *belief* (posterior probability distribution $\hat{p}(\theta)$) over the unknown reward parameters θ of the arms.
        2. At each step, *sample* a possible set of parameters $\tilde{\theta}$ from this belief: $\tilde{\theta} \sim \hat{p}(\theta)$.
        3. Choose the action that *would be best* if $\tilde{\theta}$ were the true parameters.
        4. Observe the reward and update the belief $\hat{p}(\theta)$ using Bayesian methods.
      - **Intuition:** Naturally balances exploration/exploitation. Actions likely to be optimal are chosen often, but uncertain actions still get chances proportional to the belief that they *might* be optimal. Often performs very well empirically.
- **Exploration in Deep RL (Bootstrapped DQN)**
  - **Challenge:** How to apply Thompson Sampling when the "parameters" are the weights of a complex Deep Q-Network (DQN)? Maintaining a full probability distribution over network weights is hard.
  - **Solution: Bootstrapped DQN** (inspired by Thompson Sampling):
    - **Bootstrap:** A statistical method. Create multiple (K) training datasets by sampling *with replacement* from the main replay buffer $\mathcal{D}$.
    - **Multiple Heads:** Train K different Q-value "heads" ($Q_1, \ldots, Q_K$), often sharing lower network layers. Each head Q_k is trained primarily on its corresponding bootstrapped dataset $\mathcal{D}_k$. This ensemble approximates the belief over possible Q-functions.
    - **Algorithm:**

1. At the *start* of each episode, randomly pick one head k (e.g., $k \sim \text{Uniform}\{1, \ldots, K\}$).
2. For the *entire episode*, act **consistently** based *only* on the chosen head: $a_t = \underset{a}{\text{argmax}}\, Q_k(s_t, a)$.
3. Store the experiences $(s_t, a_t, r_{t+1}, s_{t+1})$ in the replay buffer.
4. During training, update each head Q_k using data, often guided by which data corresponds to its bootstrap sample (using masks).

– **Why it Works:**

* **Deep and Temporally Consistent Exploration:** By committing to one (randomly chosen) strategy Q_k for a whole episode, the agent explores *coherently*. It's more likely to follow through on long, potentially rewarding sequences compared to the random, step-by-step exploration of ϵ-greedy.
* **Diversity:** The different heads learn diverse strategies and diverse hypothesis about the enviroment, leading to varied exploration over time.

– **Result:** Bootstrapped DQN significantly outperforms standard DQN on hard exploration games like **Montezuma's Revenge** without needing explicit reward bonuses.