Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 20: Exploration Methods

Summarized By: Behnia Soleymani

- **Recap:**
  - Reinforcement Learning agents learn through interaction, aiming to maximize rewards. However, their efficacy is fundamentally challenged in environments where rewards are sparse or delayed, a common characteristic of complex, realistic tasks. The classic comparison between the readily learnable **Breakout** (with its **dense** rewards) and the historically challenging **Montezuma's Revenge** (requiring long, unrewarded action sequences) perfectly illustrates this point. Naive exploration strategies, like **epsilon-greedy**, which rely on occasional random actions, are statistically unlikely to uncover the intricate, **temporally extended tasks** needed to succeed in such **sparse-reward** settings. The probability of executing a long, specific sequence correctly and *then* taking the right exploratory action diminishes exponentially with the sequence length. This necessitates more sophisticated approaches that go beyond simple randomness, addressing the core **exploration-exploitation dilemma** by intelligently seeking out the unknown.
- **Intrinsic Motivation: Creating Internal Drive**
  - For complex environments modeled as Markov Decision Processes (MDPs), where the sheer size or continuous nature of the state space makes visiting the exact same state twice improbable, the central strategy is **intrinsic motivation**. Instead of relying solely on the external reward $r(s, a)$, the agent is driven by a modified reward signal: $r^+(s, a) = r(s, a) + B(s, a)$. Here, $B(s, a)$ is an **exploration bonus**, an internal reward generated by the agent itself. This bonus is designed to be high for state-action pairs deemed "novel" or associated with high "uncertainty," effectively providing a dense learning signal that encourages the agent to venture into less-understood parts of the environment, even when external rewards are absent. The critical challenge then becomes designing effective mechanisms to quantify this novelty or uncertainty to calculate $B(s, a)$.
- **Quantifying Novelty: Diverse Mechanisms in Practice**
  - **Density Estimation (Pseudo-Counts):** This approach conceptualizes novelty based on the statistical likelihood of encountering a state given past experience. Practically, a **density model** (p_θ) is trained on the states D visited so far. When the agent encounters a new state s_t, this model is queried to obtain $p_\theta(s_t)$. A low probability signifies that states similar to s_t are rare in the agent's history, indicating novelty. This probability is then mathematically converted into a **pseudo-count** $\hat{N}(s_t)$ (where low probability corresponds to a low count). The bonus is typically inversely related to this count, often $B(s_t) \propto 1/\sqrt{\hat{N}(s_t)}$, rewarding visits to low-density regions. This requires maintaining and periodically retraining the density model p_θ, which can be computationally demanding. (E.g., **CTS model**).
  - **Hashing (Discrete Pseudo-Counts):** As an alternative to potentially complex density modeling, hashing maps the high-dimensional state s onto a discrete **hash code** $\phi(s)$. The core idea is that a well-designed hash function ϕ (which can be fixed or learned) groups similar states under the same code. The system then simply maintains **counts** $N(\text{code})$ for each hash code encountered. Upon visiting state s_t and computing its code $\text{code}_t = \phi(s_t)$, the current count $N(\text{code}_t)$ is retrieved. A low count implies that states mapping to this code are infrequent and thus novel. The bonus $B(s_t)$ is calculated based on this count (e.g., $B(s_t) \propto 1/\sqrt{N(\text{code}_t)}$), and the count is subsequently incremented. Its effectiveness hinges on the quality of the hash function in capturing relevant state

similarities (e.g., learned hashes in **Tang et al.**).

- **Discriminative Novelty (Exemplar Models - EX2):** This method reframes novelty not through density but through **distinguishability**. A state s_t is considered novel if it is easily differentiated from previously seen states D. Practically, an **amortized classifier network** D_{net} is trained. Given the current state s_t and a batch of distractor states $\{s_j\}$ sampled from the history buffer D, D_{net} learns to identify s_t. The novelty signal is then derived from how confidently the trained classifier identifies s_t *when compared against itself*, $p_{\text{novelty}} = D_{\text{net}}(s_t, s_t)$. High confidence implies s_t stood out from the distractors D, indicating novelty. The bonus $B(s_t)$ is calculated as an increasing function of this confidence score. This approach cleverly uses discriminative learning to implicitly gauge similarity and novelty.
- **Prediction Error (Heuristic Novelty - RND):** This popular and often highly effective heuristic equates novelty with **surprise**, measured as the error in predicting some aspect of the environment. The leading variant, **Random Network Distillation (RND)**, employs two networks: a fixed, randomly initialized **target network** f_ϕ and a **predictor network** f_θ trained to mimic the target's output. When the agent visits state s_t, both networks produce outputs: target $= f_\phi(s_t)$ and prediction $= f_\theta(s_t)$. The **prediction error**, $B(s_t) = \|\text{prediction} - \text{target}\|^2$, serves directly as the exploration bonus. The predictor network f_θ is subsequently trained (on batches of states) to minimize this error for the states it sees. Because f_ϕ is fixed and random, f_θ can only learn to predict accurately for familiar states; novel states inevitably result in high prediction errors (high bonuses), driving exploration towards surprising areas.

- **Go-Explore: Remembering and Returning for Hard Exploration**
  - Specifically designed for **hard-exploration problems** with extremely **sparse** or deceptive rewards (e.g., Montezuma's Revenge, Pitfall!), Go-Explore introduces a powerful strategy based on explicitly **remembering promising states** and **intentionally returning** to them to explore further. This approach effectively **decouples** the challenge of discovering rare states from learning a final policy, operating through two distinct phases.
  - **Phase 1: Explore and Build the Archive**: This initial phase is dedicated purely to discovery, aiming to find as many *distinct* and high-performing states as possible. It relies on maintaining an **archive** which maps a compressed state representation (**cells**) to the best-performing state found within that cell and its associated trajectory. Cells can be derived from various features like downscaled images or object coordinates. The core exploration loop proceeds as follows:

    (i) Selecting a promising cell from the archive based on criteria like score or novelty.
    (ii) **(Go)** Reliably returning the environment to the specific, high-performing state saved for that cell, which often necessitates **simulator state resetting**.
    (iii) **(Explore)** Performing exploration (e.g., random actions) from that restored state for a limited duration.
    (iv) Updating the archive if this exploration yields a superior result (e.g., higher score for an existing cell, or discovery of a state in a new cell).

    The outcome of this phase is an archive rich with diverse states, including potentially very high-reward ones, and the exact trajectories required to reach them under simulation conditions.

- **Phase 2: Robustification**: Following exploration, this phase aims to distill the knowledge gained into a usable policy. The goal is to train a robust controller (e.g., a neural network) that can reliably reach the high-reward states identified in Phase 1, crucially *without* relying on the state-resetting capability. This is typically accomplished by leveraging the best trajectories stored in the archive as demonstrations, training the policy via techniques such as **Imitation Learning** (like Behavioral Cloning) or using the trajectories to initialize or guide standard **Reinforcement Learning** algorithms, ultimately producing a policy intended for standard environment interaction.
- **Strengths and Limitations**: Go-Explore's primary strength lies in its proven ability to conquer previously intractable **hard-exploration** benchmarks by systematically searching the state space and **avoiding forgetting** how to reach rare but valuable states. Its focus on **novel state discovery** via cell representations is key to this success. However, the method faces significant practical challenges. Its heavy **dependence on simulator state-resetting** capabilities limits its applicability outside of controlled simulation environments. Furthermore, the overall performance is highly sensitive to the quality and design of the **cell representation**, and the **robustification phase** itself can be difficult, as translating a specific, high-performing trajectory from the archive into a generally reliable policy remains a non-trivial learning problem.