Artificial Intelligence and Machine Learning

Cliffwalking

Authors:
Alessandro Rocchi
Onorio Iacobelli

Contents

1 Introduction

In this Machine Learning project we will focus on the subject of Reinforcement Learning applied to the "Cliff Walking" environment. The goal of the agent is to reach a specified location, by moving in the four directions without falling down the cliff. This problem represents a fundamental challenge in Reinforcement Learning, combining elements of exploration/exploitation tradeoff, temporal credit assignment and policy optimization. This environment provides an excellent testbed for comparing different Reinforcement Learning approaches, in particular tabular Q-learning and Deep Q-Networks (DQN).

Our implementation explores both classical tabular methods, suitable for discrete state spaces and modern deep learning approaches that can handle more complex scenarios. Trough extensive experimentation and hyperparameter tuning, we demonstrate how different choices and configurations impact learning performance and convergence behavior.

2 Reinforcement Learning

Reinforcement Learning is a branch of machine learning focused on decision making trough interaction with the environment. Unlike supervised learning, where the correct outputs

are given for training, in Reinforcement Learning the agent must discover the optimal behavior by trial and error, guided only by rewards. Formally, Reinforcement Learning is often modeled as a Markov Decision Process (MDP), defined by states, actions, transition probabilities and rewards. The agent's goal is to learn a policy, a mapping from states to actions, that maximizes the expected cumulative reward over time.

The core components of a Markov Decision Process are:

- **Agent**: the learner or decision-maker that interacts with the environment. In our case this is the entity navigating the cliff walking grid.
- **Environment**: the external system that the agent interacts with. For cliff walking, this includes the grid world, cliff locations, and reward structure.
- **State (S)**: a representation of the current situation in the environment. In cliff walking, this is the agent's current position in the grid.
- **Action (A)**: the choices available to the agent at any given state.
- **Reward (R)**: the feedback signal from the environment indicating how good or bad the last action was. Rewards guide the learning process.
- **Policy (π)**: the agent's strategy for choosing actions given states. This can be deterministic (always choose the same action in a given state) or stochastic (choose actions according to a probability distribution).
- $\delta(s'|s, a)$: the probability distribution over the transitions.

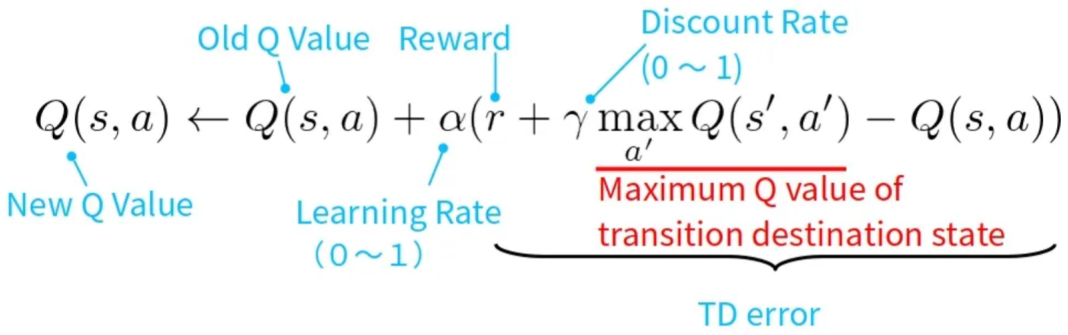
The Reinforcement Learning process follows a continuous loop:

1. Observe the current state of the environment
2. Select an action based on the current policy
3. Execute the action in the environment
4. Receive a reward and observe the new state
5. Update the policy based on the experience
6. Repeat until learning converges or a stopping criterion is met

One of the fundamental challenges in Reinforcement Learning is balancing **exploration** (trying new actions to discover potentially better strategies) with **exploitation** (using known good actions to maximize rewards). This balancing is crucial because too much exploration can lead to the agent wasting time on suboptimal actions, on the other hand with too little exploration the agent may get stuck in a local optimum and never discover the best strategy.

2.1 Q-Learning

Q-Learning is a model-free Reinforcement Learning algorithm that learns the quality of actions, telling the agent what actions to take under what circumstances. The "Q" stands for quality, representing the expected future reward for taking a specific action in a specific state.
The **Q-function** Q(s,a) represents the expected cumulative reward when taking action 'a' in state 's' and then following the optimal policy. The learning process involves iteratively updating these Q-values using the Bellman equation:



Where:

- α **(alpha)**: learning rate controlling how quickly new information overrides old information
- **r**: immediate reward received
- γ **(gamma)**: discount factor determining the importance of future rewards
- **s'**: next state after taking action 'a' in state 's'

Tabular Q-Learning maintains explicit Q-values for every (state, actions) pair in a lookup table. This approach works well for environments with small, discrete state and action spaces but becomes impractical as the state space grows.

Deep Q-Networks (DQN) use neural networks to approximate the Q-function, enabling the handling of large or continuous state spaces. Instead of storing individual Q-values, the network learns to map states to Q-values for all possible actions.

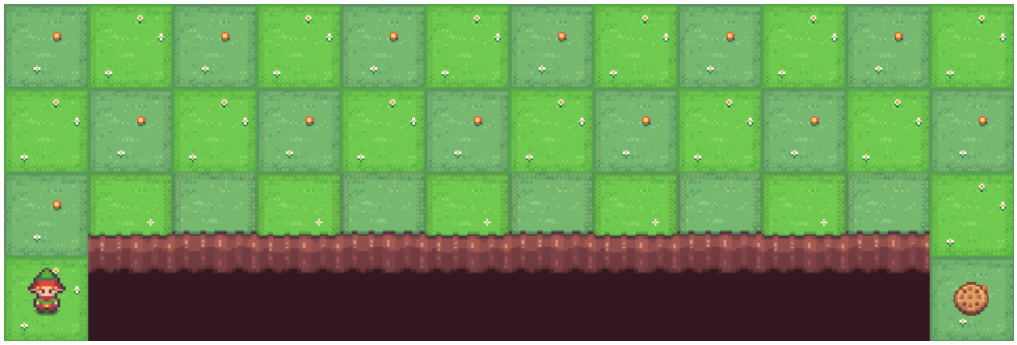
To balance exploration and exploitation, we implement an epsilon-greedy policy:

- With probability ε **(epsilon)**: choose a random action (exploration)
- With probability **(1-**ε**)**: choose the action with the highest Q-value (exploitation)

The epsilon value starts high (encouraging exploration) and gradually decreases (shifting toward exploitation) as learning progresses.

3 Environment

The Cliff Walking environment is part of the Toy Text environments which contains general information about the environment. It involves crossing a grid world from start to goal while avoiding falling off a cliff.



3.1 Description

The game starts with the player at location [3, 0] of the 4x12 grid world with the goal located at [3, 11]. If the player reaches the goal the episode ends.

A cliff runs along [3, 1..10]. If the player moves to a cliff location it returns to the start location.

The player makes moves until they reach the goal.

3.2 Action Space

The action shape is (1,) in the range {0, 3} indicating which direction to move the player.

- 0: Move up
- 1: Move right
- 2: Move down
- 3: Move left

Actions that would move the agent outside the grid boundaries result in no movement (the agent stays in the same position).

3.3 Observation Space

There are 3 x 12 + 1 possible states. The player cannot be at the cliff, nor at the goal as the latter results in the end of the episode. What remains are all the positions of the first 3 rows plus the bottom-left cell.

The observation is a value representing the player's current position as current_row * ncols + current_col (where both the row and col start at 0).
For example, the starting position can be calculated as follows: 3 * 12 + 0 = 36.

The observation is returned as an int().

3.4 Reward

The reward system is designed to encourage finding the shortest safe path while severely penalizing dangerous moves:

- **Standard step**: -1 reward (encourages shorter paths)
- **Falling off cliff**: -100 reward + episode termination
- **Reaching goal**: 0 reward + episode termination

This reward structure creates an interesting dilemma: the shortest path (along the cliff edge) is risky but potentially more rewarding if executed perfectly, while the safe path (around the cliff) is longer but more reliable.

The episode terminates when the Agent reaches the goal position, falls of the cliff or the episode can exceeds maximum step limit.

4 Models Adopted