WiDS Kalman Filtered Trend Trader

Assignment 3

Lavanya Padole 24b1292

February 2, 2026

1 Task 1: Environment Setup

The `FrozenLake-v1` environment from the `gymnasium` library was successfully initialized. The environment contains a discrete state space with 16 states corresponding to positions on a 4×4 grid and a discrete action space with 4 possible actions: left, down, right, and up.

Upon resetting the environment, a valid initial state was returned, confirming correct installation and functioning of the reinforcement learning environment. This verified that the setup was ready for subsequent reinforcement learning experiments.

2 Task 2: Tabular Q-Learning on FrozenLake

Tabular Q-learning was implemented to solve the `FrozenLake-v1` environment using an ϵ-greedy exploration policy.

2.1 Final Hyperparameters

The following hyperparameters were used for training:

- Learning rate (α): 0.1
- Discount factor (γ): 0.99
- Initial exploration rate (ϵ): 1.0
- Minimum exploration rate: 0.01
- Exploration decay rate: 0.999
- Number of episodes: 20,000

2.2 Results and Observations

At the beginning of training, the agent exhibited near-zero success due to random exploration. As training progressed and the exploration rate decayed, the agent gradually learned a policy that increased the probability of reaching the goal state.

After training, the exploration rate converged to the minimum value of 0.01, and the agent achieved an average success rate of approximately 68% over the final episodes. Perfect performance was not achieved due to the stochastic nature of the FrozenLake environment, where state transitions are probabilistic even under optimal action selection.

These results demonstrate that tabular Q-learning is capable of learning a meaningful policy in environments with discrete state and action spaces, though performance is bounded by environmental randomness.

3 Task 3: Deep Q-Learning for MountainCar

3.1 Limitations of Tabular Q-Learning

The `MountainCar-v0` environment has a continuous state space consisting of position and velocity. Tabular Q-learning is not well-suited for such environments, as discretizing the state space leads to a rapid increase in the size of the Q-table and poor generalization.

3.2 Deep Q-Network (DQN) Approach

To address this limitation, a Deep Q-Network (DQN) was implemented. In this approach, a neural network is used to approximate the Q-function, enabling the agent to generalize across continuous state spaces.

The DQN architecture consists of fully connected layers with ReLU activations and outputs Q-values corresponding to each discrete action. Experience replay is used to decorrelate training samples, and a target network is employed to stabilize training.

3.3 Training Behavior

During training, the agent initially selects actions randomly due to high exploration. As training progresses and the exploration rate decays, the agent begins to learn strategies that allow it to build momentum and climb the slope toward the goal state.

Although the agent does not achieve the goal in early episodes, gradual improvements in episode reward indicate learning progress. This behavior is consistent with typical DQN training dynamics in the MountainCar environment, where learning is incremental and requires many episodes to converge.

3.4 Discussion

The DQN-based approach demonstrates how neural networks enable reinforcement learning agents to handle continuous state spaces more effectively than tabular methods. While convergence may be slow and sensitive to hyperparameter choices, DQN provides a scalable solution for complex environments that are infeasible for traditional tabular Q-learning.

4 Conclusion

In this assignment, reinforcement learning techniques were applied to progressively more challenging environments. Tabular Q-learning proved effective for discrete environments such as FrozenLake, while Deep Q-Learning was shown to be a more suitable approach for continuous state spaces like MountainCar.

These experiments highlight the importance of selecting appropriate learning algorithms based on environment characteristics and demonstrate the strengths and limitations of both tabular and deep reinforcement learning methods.