Reinforcement learning

VOLCANO CROSSING PROBLEM

CS-414-Artifical Intelligence

Assignment no.4

**Group Members:**

Zakir Matloob (04072113055)

Muhammad Junaid Fida (04072113048)

**Submitted To: Dr. Ayaz Hussain**

**Reinforcement Learning for Volcano Crossing Problem: Algorithm Implementation and Analysis Report**

**Introduction:**

In this report, we present the implementation and analysis of three RL algorithms Model-Free Monte Carlo, Q-Learning, and SARSA applied to the Volcano Crossing problem. The problem involves a grid-world environment where an agent needs to navigate from the start state to one of the end states while avoiding dangerous states.

**Problem Definition:**

**Grid Size and States:**

The grid size is defined as (4, 3), and the end states are located at (1, 2) and (3, 1). Dangerous states, where negative rewards are encountered, are specified at (1, 1) and (2, 2).

**Slip Probability, Epsilon, and Gamma:**

The slip probability, denoting the chance of the agent slipping and taking an unintended action, is set to 0.1. Epsilon is used for epsilon-greedy policies, with a value of 0.1. Gamma, the discount factor for future rewards, is set to 0.9.

**Model-Free Monte Carlo:**

The Model-Free Monte Carlo algorithm is implemented for learning the optimal policy. The agent explores the environment, and episodes are generated using an epsilon-greedy policy. Slip probability is incorporated to simulate uncertainties in the agent's actions. The Q-values are updated based on the observed returns, and the algorithm converges over a specified number of episodes.

**Results and Analysis:**

The Model Free Monte Carlo algorithm was executed with the following parameter configuration:

* Slip Probability: 0.1
* Epsilon: 0.1
* Number of Episodes: 100

The Model-Free Monte Carlo algorithm was employed to train an agent for the Volcano Crossing problem. The slip probability was set to 0.1, introducing a level of uncertainty in the agent's actions. The exploration-exploitation trade-off was controlled by epsilon, with a value of 0.1. The algorithm's performance was evaluated over 100 episodes. The average utility values for each episode provide insights into the learning progress. Here are some key observations:

* + **Episodes: 1, Average Utility: 0.80**
  + **Episodes: 10, Average Utility: 10.53**
  + **Episodes: 20, Average Utility: 11.75**
  + **Episodes: 30, Average Utility: 12.09**
  + **Episodes: 40, Average Utility: 12.32**
  + **Episodes: 50, Average Utility: 13.12**
  + **Episodes: 60, Average Utility: 13.33**
  + **Episodes: 70, Average Utility: 13.35**
  + **Episodes: 80, Average Utility: 13.39**
  + **Episodes: 90, Average Utility: 13.41**
  + **Episodes: 100, Average Utility: 13.43**

The average utility steadily increased, indicating that the agent improved its decision-making over time.

**Q-Values:**

The Q-values represent the learned values for each state-action pair. For instance:

* **State (0, 0): Q-values - [3.91, 11.26, 3.65, 8.41]**
* **State (1, 1): Q-values - [-0.04, -11.82, 8.95, 20.00]**
* **State (2, 2): Q-values - [20.00, -28.80, 18.00, -32.90]**
* **State (3, 0): Q-values - [14.53, 17.50, 17.54, 20.00]**

These values represent the agent's learned preferences for actions in different states.

**Utility Matrix:**

The utility matrix illustrates the expected utility for each state:

* + **State (0, 0): Utility - 11.26**
  + **State (1, 1): Utility - 20.00**
  + **State (2, 2): Utility - 20.00**
  + **State (3, 0): Utility - 20.00**

Higher utility values indicate more favorable outcomes.

The Model-Free Monte Carlo algorithm successfully learned a policy for the Volcano Crossing problem. The training process resulted in improved average utilities, and the Q-values and utility matrix provide valuable insights into the agent's decision-making capabilities. The slip probability of 0.1 introduced a level of uncertainty, and the agent adapted its strategy accordingly.

**Q-Learning:**

Q-Learning is employed to learn the optimal action-value function. The agent iteratively updates Q-values using the Q-learning update rule. Exploration is facilitated by an epsilon-greedy strategy, and slip probability is considered. The algorithm converges as the Q-values stabilize.

**Results and Analysis**

The Q-Learning algorithm was executed with the following parameter configuration:

* Epsilon: 0.01
* Number of Episodes: 500
* Slip Probability: 0.2

After running the Q-Learning algorithm for **500** episodes, the learned **Q-values** for each state-action pair in the Volcano Crossing problem are as follows:

**[[[ 14.58 13.122 14.58 16.2 ]**

**[ 16.2 -32. 14.58 18. ]**

**[ 18. 20. 16.2 18. ]]**

**[[ 14.58 3.29615177 12.15445754 -29.25132592]**

**[ 14.46053825 12.10515739 11.98588422 20. ]**

**[ 0. 0. 0. 0. ]]**

**[[ 8.82217315 0. 1.1526337 0.72 ]**

**[ -6.400196 18.28201308 0. -17.28 ]**

**[ 4. 0. 0.72 0. ]]**

**[[ 0. 0. 0. 0. ]**

**[ 0. 0. 0. 0. ]**

**[ 0. 0. 0. 0. ]]]**

The utility values represent the maximum Q-value for each state. The utility matrix is as follows:

**Utility Values**

**[[16.2 18. 20. ]**

**[14.58 20. 0. ]**

**[ 8.82217315 18.28201308 4. ]**

**[ 0. 0. 0. ]]**

The Q-values and utility values show what the agent has learned about making decisions in different situations. The slip probability of 0.2 adds some randomness, and the agent learns to move through the game, taking into account the chance of slipping. Further experimentation with different parameters and comparisons could enhance our understanding of the learning process and lead to potential improvements.

**SARSA:**

The State-Action-Reward-State-Action (SARSA) algorithm is utilized for learning the optimal policy. The agent interacts with the environment, and Q-values are updated based on observed rewards and next-state actions. Epsilon-greedy exploration and slip probability are integrated, allowing the agent to learn the optimal policy over multiple episodes.

**Results and Analysis**

The SARSA algorithm was executed with the following parameter configuration:

* Epsilon: 0.01
* Number of Episodes: 1000
* Slip Probability: 0.1

After running the SARSA algorithm for 1000 episodes, the average utility achieved was **-55.35.** This value provides an indication of the overall performance of the learned policy. Negative values are expected due to the presence of negative rewards associated with dangerous states.

The Q-values represent the learned values for state-action pairs. The following matrix illustrates the Q-values for each state-action pair in the Volcano Crossing problem:

**Q-values:**

**[[[-1.08327863e+01 -7.62046423e+01 -1.66469561e+01 1.68220561e-02]**

**[-2.95276983e+00 -1.06698600e+02 -6.58761344e+00 1.02544674e+01]**

**[ 1.68754577e+01 2.00000000e+01 -5.91221176e+01 1.77545002e+01]]**

**[[-1.45140213e+01 1.40033128e+01 6.38943042e+00 -1.55696417e+02]**

**[ 2.96947390e+00 -1.16670727e+02 -1.08557742e+02 1.91203907e+01]**

**[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]**

**[[ 1.17396239e+01 1.63173906e+01 1.32834786e+01 2.46272262e+00]**

**[-1.32125390e+02 1.99999979e+01 8.00795798e-01 -1.33133472e+02]**

**[ 1.82820131e+01 -1.07518367e+02 -1.18568532e+02 -2.51381187e+01]]**

**[[ 6.30135060e+00 1.79998956e+01 1.79885897e+01 2.00000000e+01]**

**[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]**

**[-1.38226398e+02 7.56955238e+00 1.78525164e+01 5.64660818e+00]]]**

**Utility values:**

The utility values represent the maximum Q-value for each state. The utility matrix is as follows:

**[[1.68220561e-02 1.02544674e+01 2.00000000e+01]**

**[1.40033128e+01 1.91203907e+01 0.00000000e+00]**

**[1.63173906e+01 1.99999979e+01 1.82820131e+01]**

**[2.00000000e+01 0.00000000e+00 1.78525164e+01]]**

The SARSA algorithm successfully learned a policy for the Volcano Crossing problem, as evidenced by the achieved average utility and the Q-values. The negative average utility reflects the challenges introduced by dangerous states, emphasizing the agent's ability to navigate through the environment while considering potential risks.

**Conclusions:**

* We tested the algorithms using different numbers of episodes under a random policy. We tracked Q-values and average utility over time. As episodes increased, Q-values converged, showing that the algorithms learned the best strategy.
* We explored slip probabilities from 0.0 to 0.3. Higher slips made the agent's actions more uncertain. The results revealed that higher slip probabilities made learning harder, needing more episodes for the system to adapt.
* We used epsilon-greedy policies for exploration, where epsilon balanced exploration and exploitation. Different epsilon values were tested, with higher values favoring exploration and lower values focusing on exploitation.