Machine Learning Lab 3 Report: Reinforcement learning

Luis Calderón Robustillo, Andrés Martínez Silva

December 2, 2023

This report is divided into two main sections which cover the main topics of this lab: Markov Decission Processes (MDPs) and Reinforcement Learning.

1 MDPs

MDPs are non-deterministic search processes where there is a known model of the environment composed of states, actions, transitions and discounted rewards. Each step, the agent takes an action and receives a reward. Solving an MDP implies finding a map of states to actions (policy) that maximizes the sum of discounted rewards (utility) for each of the states [1].

Exercise 1: Value Iteration

Value iteration is an offline planning algorithm for solving MDPs where we compute k-step estimates of the optimal values for each state. It's an iterative process where the computed values eventually converge to the optimal ones after an infinite number of iterations. In this case, the algorithm was run for 100 iterations, which is enough for convergence in GridWorld.

$$V_{k+1}(x) = \max_a \sum_{x \in X} p(x'|x,a)(R + \gamma V_k(x')) \quad (1)$$

Exercise 2. Bridge crossing analysis

In this exercise, the discount rate (γ) and noise (n) parameters are tuned individually so that the resulting policy makes the agent cross the bridge. First, we tried reducing γ while keeping the default n=0.2 fixed with no success. Then, we fixed the default $\gamma = 0.9$ and reduced n until we found a valid solution for n = 0.01. Due to the high negative reward of falling off the bridge, the agent would only risk crossing it if there was a very low probability of taking an action forward and falling off due to noise. On any other case, the resulting policy would always make the agent take the safe exit to the left.

Exercise 3. Policies

In this exercise, we tune the discount rate(γ), noise (n) and living reward (r) to get optimal policies which exhibit specific behaviors from the agent. In these experiments we found out the following: (1) reducing both n and r was critical to make the agent risk the dangerous route over the safer one -the agent prioritizes staying alive and only risks the cliff if there is a very small probability of falling off-, and (2) γ must be greatly reduced to make the agent take the close exit over the distant one -meaning rewards later in the future have very little importance. However, with $\gamma = 1$ (no discount) the agent will never reach any of the terminal states.

case	γ	n	r
close exit, risk cliff	0.25	0.001	0.05
close exit, safe route	0.1	0.1	0.25
distant exit, risk cliff	0.8	0.05	0.25
distant exit, safe route	0.9	0.2	0.45
avoid exits and cliff	1.0	0.2	0.45

2 Reinforcement learning

While the objective is still to optimize behavior in an MDP, in reinforcement learning the agent does not know how the world works. This means there is

not a transition or rewards model available and the agent must experiment and try out actions and states to learn in order to find the best policy [2].

Exercise 6. Q-Learning

Q-learning is a sample-based Q-value iteration algorithm where the agent learns and updates estimates of Q-values based on experiences, as it takes actions and explores the environment, where each estimate is incorporated into a running average limited by a learning rate α.

$$Q(s,a) = Q(s,a) + \alpha \left(R + \gamma \max_a Q(s',a') - Q(s,a) \right) \tag{2}$$

Challenges emerged near negative reward areas due to environmental noise, indicating the need of a high number iterations to achieve a good result. Despite this, the agent showcases adaptability and potential for improvement.

Exercise 7. Epsilon Greedy

Observing the agent's behavior, results are consistent with those obtained in value iteration, especially along well-traveled paths. Average rewards are slightly lower than the Q-values predict due to random actions and the initial learning phase. In GridWorld, the agent with $\epsilon = 0.1$ struggles initially and tends to stick to previously learned paths. The agent with $\epsilon = 0.9$ explores extensively but may neglect optimal paths after discovery. This way, lower values promote focused learning but risk convergence to suboptimal paths, while a higher ones accelerate exploration but may lead to overlooking optimal solutions. A careful tuning of ϵ is crucial for balancing exploration and exploitation in the learning process.

Exercise 8. Bridge crossing revisited

In this exercise, achieving the optimal policy within 50 iterations on the noiseless BridgeGrid is not possible due to the challenge of obtaining a high reward without encountering negative rewards along the bridge, regardless of epsilon and learning rate settings. Even with high values of epsilon and learning rate we would need more iterations to cross the bridge.

Exercise 9. Q-learning and Pacman

On SmallGrid, Pacman reliably wins test games after learning, demonstrating the effectiveness of Q-learning for low-dimensional state spaces. As in previous exercises, changing ϵ can vary the final result because of trying more random actions. However, the method struggles on mediumGrid, indicating limitations in scaling due to separate Q-values for each board configuration. Scalability issues suggest the need for improved methods to handle larger state spaces.

Exercise 10. (Optional) Approximate Q-learning.

For this exercise, we implemented an Approximate Q-learning agent that utilizes feature weights for state-action pairs. The weight vector, associated with specific features, is updated based on the Q-learning error. The agent uses feature extractors to capture relevant aspects of the environment. The introduction of approximate Q-learning with feature weights significantly enhances the agent's ability to scale to larger problems. By leveraging a more sophisticated representation of the environment through feature extractors, the agent performs significantly better.

Conclusions

In Section 1, we addressed a simple MDP where we ran value iteration to find the best policy, then performed parameter tuning in different scenarios to study how the discount rate, noise and living reward influence the obtained policies. In Section 2 we moved on to a reinforcement learning problem where we studied the Q-learning algorithm, which showcased effectiveness in learning optimal policies for low dimensional state spaces. However, we found out there are still some challenges, like exploring environments with many negative rewards (cross the

bridge) or finding the optimal policy in larger and more complex scenarios, highlighting the need for advanced techniques to address scalability issues.

To conclude, for the appended gaussian process optional exercise, references [3] and [4] were consulted.

References

[1] Daniel Klein Pieter Abbeel. *COMPSCI 188 - 2018-09-20 - Markov Decision Processes (MDPs)*. Youtube. 2018. URL: https://www.youtube.com/watch?v=ZToWj64rxvQ&list=PLsOUugYMBBJENfZ3XAToMsg44W7LeUVhF&index=9.

[2] Daniel Klein Pieter Abbeel. *COMPSCI 188 - 2018-09-25 - Reinforcement Learning*. Youtube. 2018. URL: https://www.youtube.com/watch?v=TiXS7vROBEg&list=PLsOUugYMBBJENfZ3XAToMsg44W7LeUVhF&index=10.

[3] Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. Chap. 2.

[4] scikit-learn. *Gaussian processes*. scikit-learn. URL: %5Curl%7Bhttps://scikit-learn.org/stable/modules/gaussian_process.html%7D. (accessed: 30.11.2023).