Using RL to create an efficient Taxi agent

Venkat Narra (5033)
Sai Siddhardha Maguluri (5033)

Abstract

We investigated the performance of Q-learning and SARSA algorithms in solving the Taxi environment, a classic grid-world problem in reinforcement learning. Both algorithms are implemented and compared in terms of their average reward over multiple episodes. The impact of different hyper-parameters on agent performance, including learning rate, and discount factor, was also examined. In addition, We explored the impact of using different policies to improve the agent's learning performance. The results suggest that the Q-learning algorithm converges faster than the SARSA algorithm in the taxi environment.

1. Domain

1.1. History

The Taxi Problem is all about navigating to passengers in a grid world, picking them up, and dropping them off at one of four locations. There are four designated pick-up and drop-off locations (Red, Green, Yellow, and Blue) in the 5x5 grid world.

1.2. Open AI Gym

In our project, we utilized the "Taxi-v3" environment `https://gymnasium.farama.org/environments/toy_text/taxi/` from OpenAI Gym to train different models.

1.3. Environment

For this problem, we have a state space of 500 since we have 25 taxi positions(5*5 grid), 5 passenger locations, and 4 destination locations and we have 6 actions that an agent can take like moving south, moving north, moving east, moving west, pickup passenger, and finally drop off a passenger. The taxi starts off at a random square in the grid and the passenger at one of the four designated locations. The main goal of this is to move the taxi to the passenger's location, pick up the passenger, move to the passenger's desired destination, and drop off the passenger. Once the passenger is dropped off we can start the whole process once again. The agent receives a reward of +20 for successfully dropping off the passenger at the correct location. A reward of -10 for incorrect attempts to pick up/drop off passengers and -1 for each step.

2. Hypotheses

Hypotheses are the following:

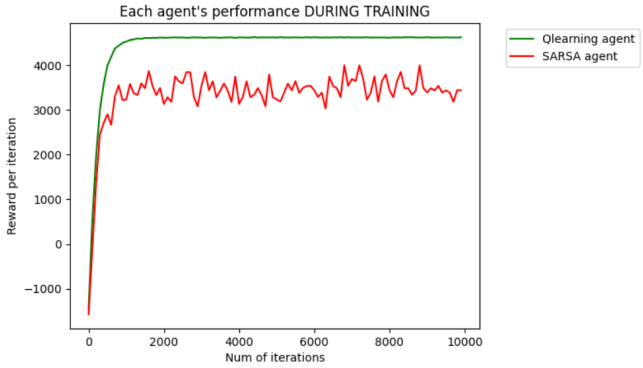
1. We hypothesize that the Q-learning algorithm will converge faster than the SARSA algorithm in the Taxi environment.
2. We hypothesize with the increase in the learning rate, the agent learns better.
3. Greedy-Epsilon policy helps the SARSA agent to learn faster in the taxi environment compared to the softmax policy.
4. After the agent learns optimal policy, regardless of the epsilon decay method(i.e Exponential decay, inverse decay, and linear decay) the reward will be constant.

3. Experiments

3.1. Hypothesis 1

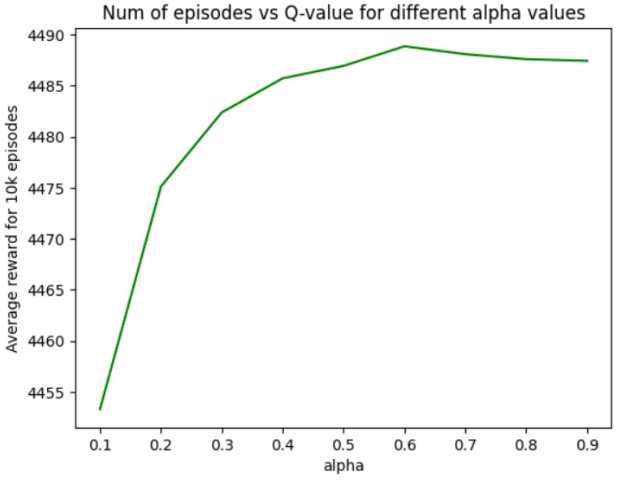
Using Reinforcement learning we can train an agent in learning about the taxi environment and navigate effectively. In a taxi environment, the untrained agent will take random actions. In below figure, we can see the untrained agent's reward is always negative.

We trained our agent with Q-learning and SARSA algorithms with hyperparameters of $\epsilon = 1.0$, $\alpha = 0.1$, $\gamma = 0.8$, and using ϵ greedy policy. we have chosen these algorithms due to their ability to learn without knowledge of the Markov Decision Process. We trained our agent for 10,000 episodes and each episode is 500 timesteps. The agent trained using the Q-learning algorithm has achieved more reward per episode compared to SARSA. From the below figure, we can see that Q-learning converges faster than SARSA in the taxi environment. Converging means learning optimal policy for 10k episodes compared to other values. Q-learning learned the optimal policy around 1800 episodes, as we can see the rewards becomes constant after this.



3.2. Hypothesis 2

The learning rate is a hyperparameter that controls the rate at which the agent updates its estimates of the value function or policy. It determines how much the agent should update its estimate of the value function or policy based on new information obtained from its experience. So, we trained our Q-Learning agent with different learning rates and we see an increase in rewards as the learning rate increased.



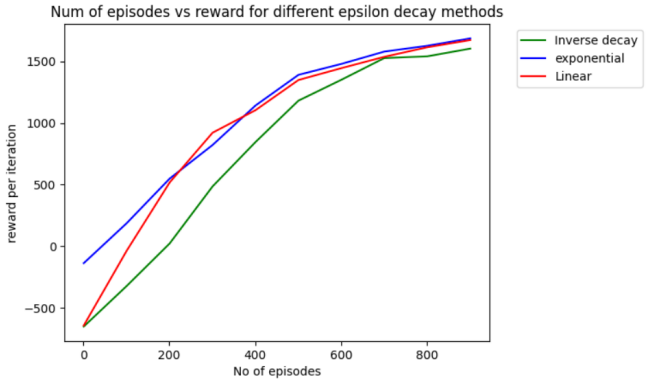
If the learning rate is too high, the agent will give more weight to the most recent experience and update its estimates more quickly. However, this can cause the agent to overreact to noisy or irrelevant information, leading to instability and poor performance. On the other hand, if α is set to a low value, the agent will give less weight to the new experience and rely more on its previous estimates, leading to slower but more stable learning. We can see a learning rate of 0.6 gave more average reward for 10k episodes compared to other values.

3.3. Hypothesis 3

A policy is a mapping from states to actions that the agent uses to make decisions. The policy defines the behavior of the agent in the environment, and its goal is to maximize the expected cumulative reward over time. The hyperparameters we used for this experiment are $\epsilon = 1.0$, $\alpha = 0.1$, $\gamma = 0.8$ and decay rate of 0.05. we trained the SARSA agent for 10k episodes, each episode is of 500 timesteps. In value-based methods like Q-learning and SARSA, a policy is a simple look-up table. We have different kinds of policies like greedy policy, ϵ-greedy policy, and softmax policy. We observed that ϵ-greedy policy gives us more reward compared to softmax policy for SARSA.

3.4. Hypothesis 4

Epsilon decay is a technique used in reinforcement learning to gradually reduce the level of exploration in an agent's behavior policy over time. Epsilon is the parameter that controls the amount of exploration versus exploitation that an agent uses when selecting its actions. We have different kinds of ϵ decay methods like linear, exponential, and inverse decay. The exponential decay of epsilon gives us more reward in the initial phase, but once the agent learns optimal policy, we observed that decay method doesn't have much effect on the reward that the agent gets.



4. Learning Methods

In this project, we implemented the Q-Learning and SARSA learning methods under Reinforcement Learning. For this taxi game, Q learning is performing better than the SARSA. We used the parameters we used are learning rate = 0.6, discount rate = 0.8, epsilon = 1.0, decay rate= 0.005, num episodes = 10000, max steps = 500.

4.1. Q-Learning

Q-learning is a value-based reinforcement learning algorithm. Here an agent is trained to produce an optimal strategy for solving a problem by trying to learn the best course of action from its history of interactions with the environment. Q*(s, a) is the expected action "a" taken at the state "s".

4.2. SARSA

The SARSA is a TD Learning approach that is on-policy. The main distinction between SARSA and Q-learning is that a new action—and therefore reward—is chosen using the same policy that decided the initial action than taking the maximum reward for the next state.

5. Literature Review

5.1. Q-Learning

The paper by Watkins and Dayan(1992) introduced the Q-learning algorithm. The Q-learning algorithm is a model-free method for learning optimal policies in Markov decision processes (MDPs). Q-learning is based on the idea of estimating the optimal action-value function, or Q-function, which gives the expected reward for taking a particular action in a particular state and following an optimal policy thereafter. We implemented the algorithm that is mentioned in Sutton, R. S. and Barto, "Reinforcement Learning: An introduction".

The paper by Thomas G. Dietterich(1999) [4] used two algorithms Q-learning and SARSA for the taxi environment and used a tabular representation of the action-value function Q(s, a).

6. Conclusion

We trained and experimented with the agent performance with two reinforcement learning methods namely Q-learning and SARSA. The agent was able to learn about the taxi environment and navigate effectively and achieved a better reward compared to an untrained agent. We have observed that with a learning rate of 0.6, the agent was able to get a maximum reward in given time steps compared to other rates. We also observed that for the taxi environment, because the state space is small, once the Q-learning agent finds the optimal policy, we got approximately the same reward irrespective of policies and epsilon decay methods used.

7. Future Work

We have trained our agent for 10k and 20k episodes, and we have observed that there are still zero values for some state-action pairs in the Q-table, which means the agent didn't explore all the states. In the future, we wish to explore a different algorithm that is proposed in [3] to improve Q-table values and increase the reward that the agent gets. Also, we would like to explore different Q-learning algorithms like Deep Q-Learning, Double Q-learning, and Hierarchical Q-Learning that are proposed in [2] for taxi environment and compare them.

8. Contributions

Venkat Narra implemented the Q-learning algorithm and performed experiments in accordance with Hypotheses 1 and 2. Sai Siddhardha Maguluri implemented the SARSA algorithm and performed experiments in accordance with Hypotheses 3 and 4.

References

1. Watkins, C.J.C.H., Dayan, P. Q-learning. Mach Learn 8, 279–292 (1992). https://doi.org/10.1007/BF00992698

2. F. Esmaeily and M. R. Keyvanpour, "WMat algorithm based on Q-Learning algorithm in taxi-v2 game," 2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT), Mashhad, Iran, 2020, pp. 112-118, doi: 10.1109/SCIOT50840.2020.9250211.

3. B. Jang, M. Kim, G. Harerimana and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," in IEEE Access, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

4. Thomas G. Dietterich(1999), "Hierarchical Reinforce-

ment Learning with the MAXQ Value Function Decomposition". https://arxiv.org/abs/cs/9905014

5. Sutton, R. S. and Barto, A. G. (eds.). Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA, 2018.