Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno:

Relatório do Laboratório 11 - Aprendizado por Reforço Livre de Modelo

1 Breve Explicação em Alto Nível da Implementação

Para ambos os métodos, foi utilizado:

```
def epsilon_greedy_action(q, state, epsilon):
        a_chc = np.random.uniform(0, 1)
        a_rng = q.shape[1]
        a_max_idx = np.argmax(q[state, :])
        if a_chc >= epsilon: # Did NOT opt for random action
                return a_max_idx
        else:
                return int(np.random.randint(0, a_rng, 1))
```

1.1 SARSA

Pela implementação proposta, bastou implementar o passo de atualização, como feito a seguir:

```
class Sarsa(RLAlgorithm):
        def __init__(self, num_states,
                num_actions, epsilon, alpha, gamma):
                super().__init__(num_states,
                        num_actions, epsilon, alpha, gamma)

        def get_greedy_action(self, state):
                a = epsilon_greedy_action(self.q, state,
                        self.epsilon)
                return a

        def learn(self, state, action, reward, next_state,
                next_action):
                obj_dist = reward
                        + self.gamma*self.q[next_state, next_action] -
                        self.q[state, action]
                self.q[state, action] = self.q[state, action] +
                        self.alpha * obj_dist
                pass
```

1.2 Q-Learning

Semelhante ao exposto para o método SARSA, fez-se:

```
class QLearning(RLAlgorithm):
        def __init__(self, num_states, num_actions,
                epsilon, alpha, gamma):
                super().__init__(num_states, num_actions,
                        epsilon, alpha, gamma)

        def get_greedy_action(self, state):
                a = epsilon_greedy_action(self.q, state,
                        self.epsilon)
                return a

        def learn(self, state, action, reward,
                next_state, next_action):
                obj_dist = reward +
                self.gamma*np.max(self.q[next_state, :]) -
                self.q[state, action]
                self.q[state, action] = self.q[state, action] +
                self.alpha*obj_dist
                pass
```

2 Figuras Comprovando Funcionamento do Código

2.1 SARSA

2.1.1 Tabela Ação-Valor e Política *Greedy* Aprendida no Teste com MDP Simples

```
[[ -9.46830364  -8.01352809 -10.18989999]
 [-10.56478603  -9.12412732 -11.33010052]
 [-10.83100042 -10.36422043 -11.63203286]
 [-11.71859057 -11.3627901  -12.10822051]
 [-12.33441422 -12.30353703 -12.26124947]
 [-11.71293995 -11.72399215 -11.47736594]
 [-11.24696982 -11.81371122 -10.25574893]
 [-10.59963408 -11.349192    -9.15626364]
 [ -9.26112936 -10.43619617  -8.12920054]
 [ -7.05923992  -8.03122254  -8.06739806]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

Figura 1: Resultado do teste, método SARSA.

2.1.2 Convergência do Retorno

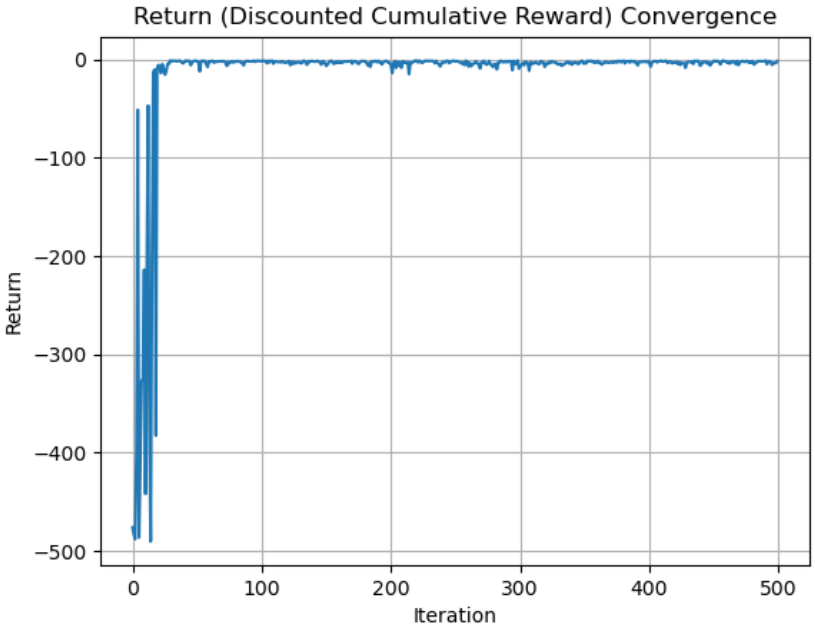

Figura 2: Convergência do retorno, método SARSA.

2.1.3 Tabela Q e Política Determinística que Seria Obtida Através de *Greedy*(Q)

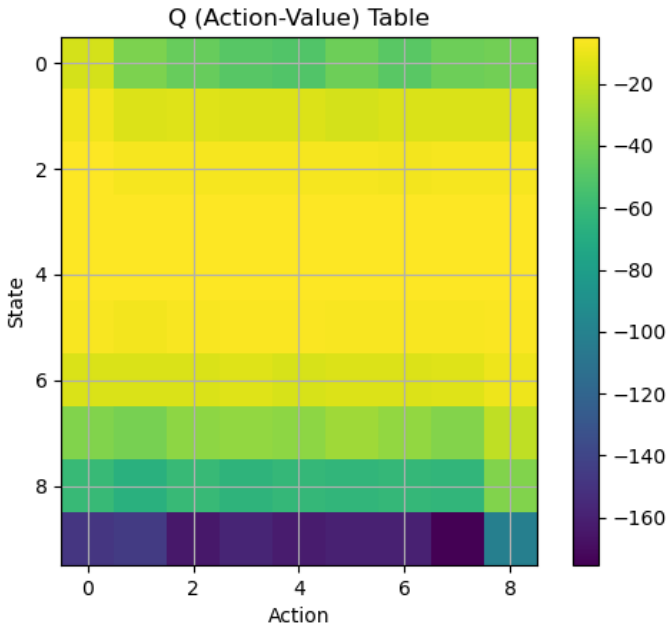

Figura 3: Tabela Q, método SARSA.

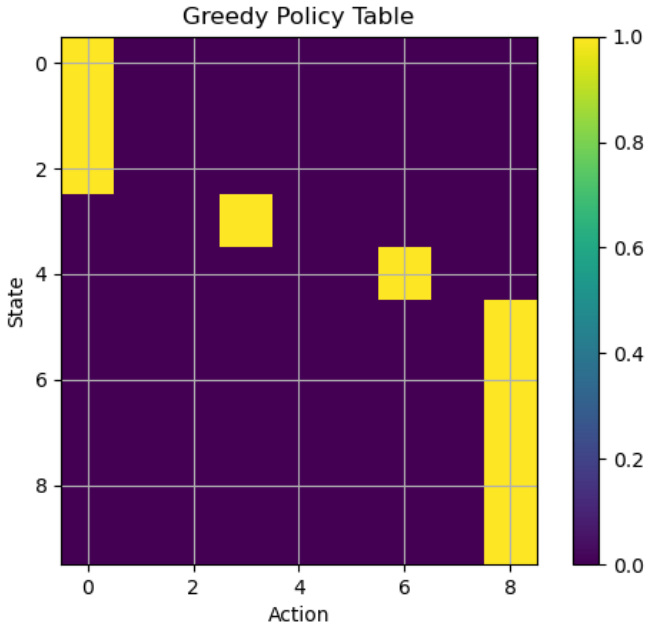

Figura 4: Política, método SARSA.

2.1.4 Melhor Trajetória Obtida Durante o Aprendizado

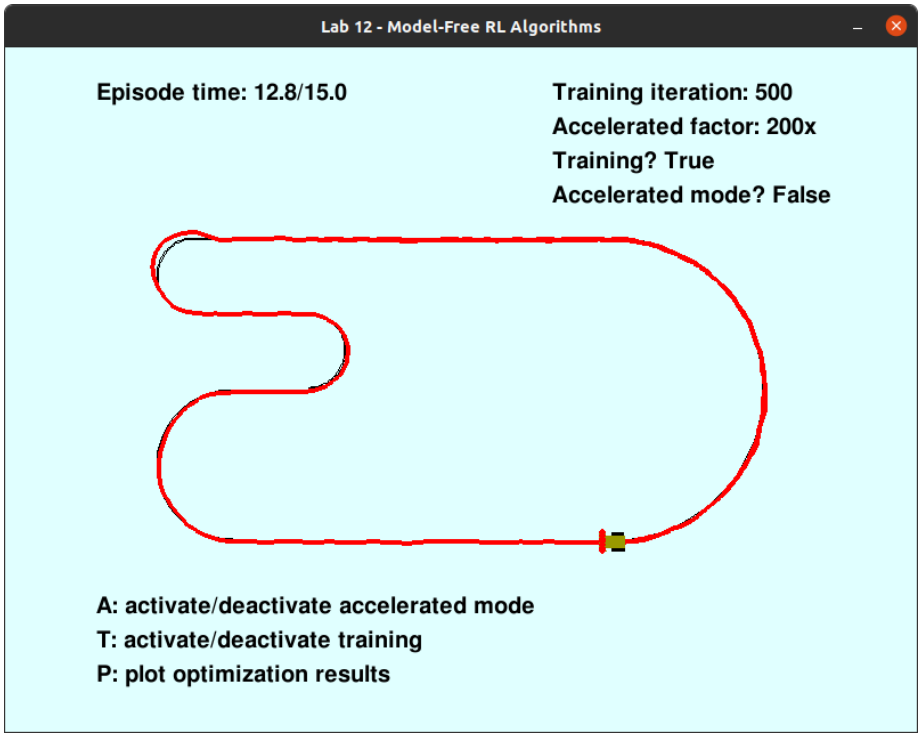

Figura 5: Solução, método SARSA.

2.2 Q-Learning

2.2.1 Tabela Ação-Valor e Política *Greedy* Aprendida no Teste com MDP Simples

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.96639265 -1.99       -3.93233622]
 [-3.70261225 -2.9701     -4.12858066]
 [-4.34514517 -3.94039894 -4.67223391]
 [-5.16982991 -4.8981436  -4.89800394]
 [-4.33181895 -4.98258077 -3.9403986 ]
 [-3.65217242 -3.79337816 -2.9701    ]
 [-2.96506248 -3.93686758 -1.99      ]
 [-1.99       -2.9701     -1.        ]
 [ 0.         -0.99       -0.99      ]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

Figura 6: Resultado do teste, método Q-Learning.

2.2.2 Convergência do Retorno

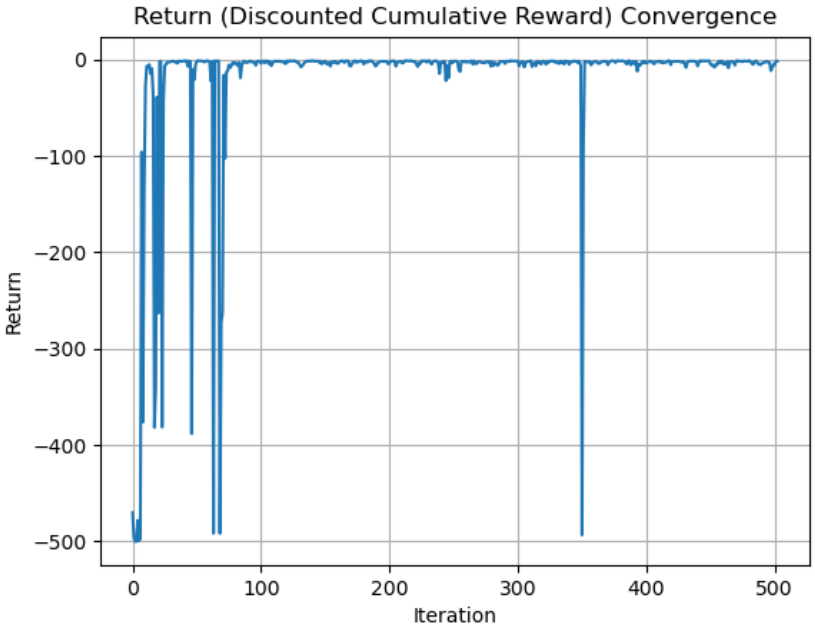

Figura 7: Convergência do retorno, método Q-Learning.

2.2.3 Tabela Q e Política Determinística que Seria Obtida Através de *Greedy*(Q)

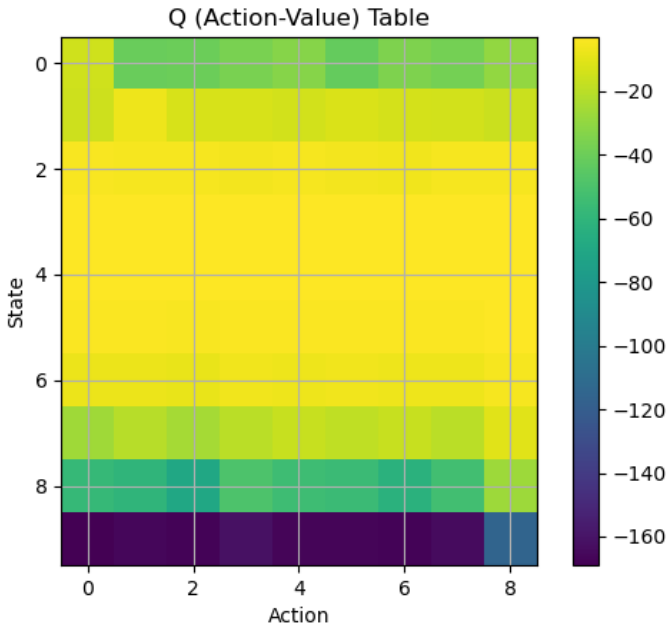

Figura 8: Tabela Q, método Q-Learning.

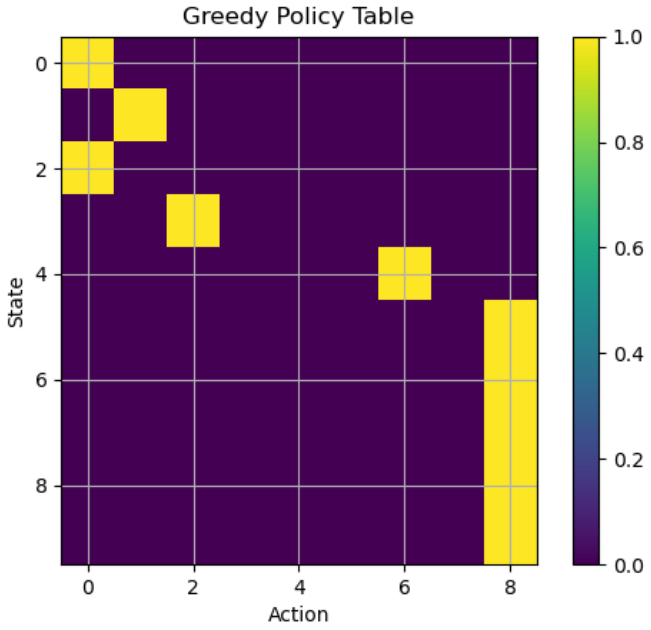

Figura 9: Política, método Q-Learning.

2.2.4 Melhor Trajetória Obtida Durante o Aprendizado

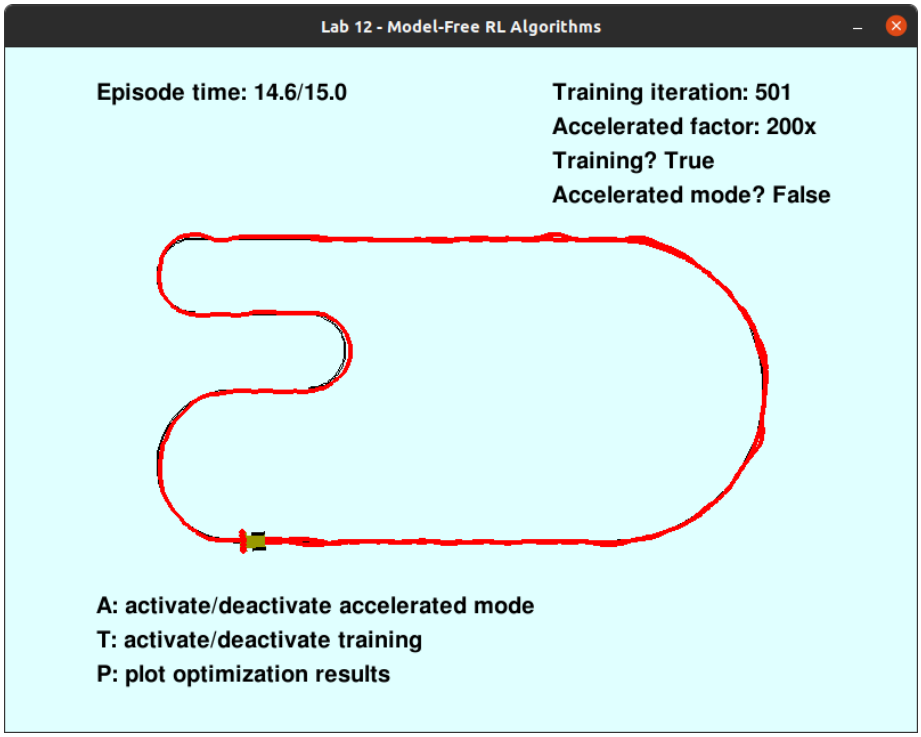

Figura 10: Solução, método Q-Learning.

3 Discussão dos Resultados

Os resultados obtidos no momento de teste seguem o esperado, dado que a solução obtida pelo SARSA se mostrou mais conservadora, enquanto aquela obtida pelo método Q-Learning é ótima.

Já para as otimizações dos trajetos, é interessante notar que ambos chegam em uma solução de custo baixo ao final, porém com o método Q-Learning convergindo mais rapidamente.

As tabelas de política se mostram semelhantes para o caso de treino, enquanto na tabela ação-valor a solução do método Q-Learning tem valores mais baixos como um todo. Isso reflete novamente a busca do método SARSA ser mais conservadora.