Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT-213
Aluno: Rafael Mello Celente

Relatório do Laboratório 11 - Aprendizado por Reforço Livre de Modelo

1. Breve Explicação em Alto Nível da Implementação

1.1. SARSA

O SARSA é um algoritmo *on-policy* TD tabular que funciona estimando a função valor e otimizando a política. Ele funciona atualizando a política baseado em 5 dados: o estado s_t, a ação a_t, a recompensa r_t, o próximo estado s_{t+1} e a próxima ação a_{t+1}.

Para a implementação neste laboratório, precisou-se implementar as funções `get_greedy_action, epsilon_greedy_action e learn`. Essas funções tiveram por finalidade escolher uma ação através de uma política ε-*greedy* e estimar a função $Q(s_{t+1}, a_{t+1})$ a partir da equação 1.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

1.2. Q-Learning

O algoritmo de Q-learning, em contrapartida, é um algoritmo *off-policy* TD tabular que funciona estimando a função valor e otimizando a política. Diferentemente do SARSA, o Q-learning, por ser *off-policy,* pode usar experiências passadas para melhorar a política.

Para a implementação neste laboratório, precisou-se implementar as funções `get_greedy_action, greedy_action e learn`, assim como o SARSA, porém alterando a função *greedy* e a equação de atualização implementada. Essas funções tiveram por finalidade escolher uma ação através de uma política *greedy* e estimar a função $Q(s_{t+1}, a_{t+1})$ a partir da equação 2. Pode-se perceber que as equações 1 e 2 são parecidas. Entretanto, a função *max* aplicada em 2 permite que o algoritmo trabalhe *off-policy*, permitindo que se aprenda com outras experiências passadas.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

2. Figuras Comprovando Funcionamento do Código

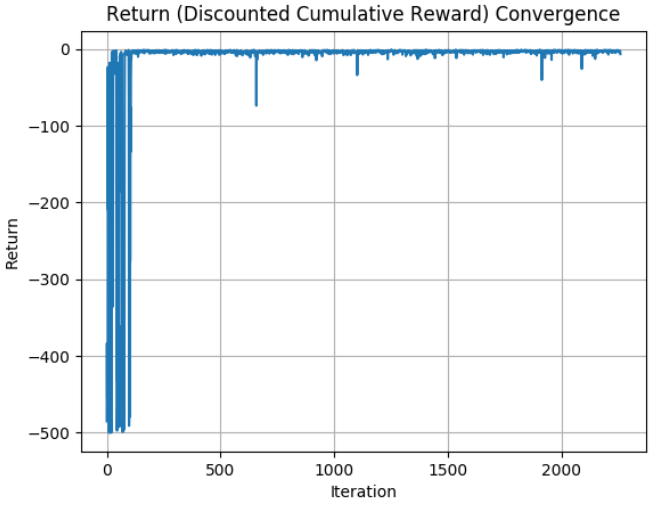
Basta colocar as figuras.

2.1. SARSA

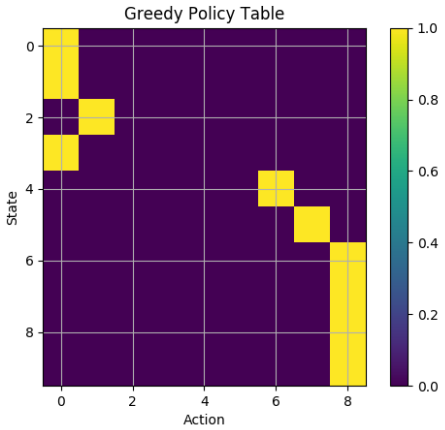
2.1.1. Tabela Ação-Valor e Política *Greedy* Aprendida no Teste com MDP Simples

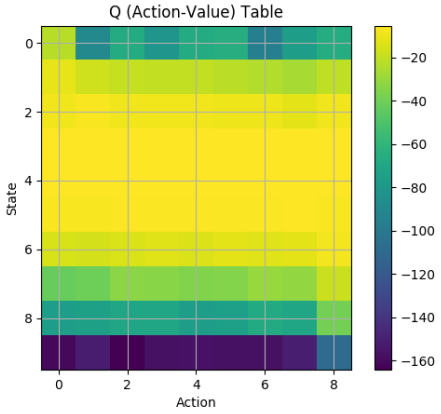
```
Action-value Table:
[[ -9.60978905  -7.78984604 -10.45525871]
 [-10.37413012  -9.30280606 -11.42661049]
 [-10.97061384 -10.46634613 -11.28390437]
 [-11.69420996 -11.39885856 -11.88610642]
 [-12.35005457 -12.22946132 -12.2232755 ]
 [-11.53094558 -11.98129181 -11.38825426]
 [-11.06750953 -11.46877644 -10.56479515]
 [-10.46374193 -11.31575505  -9.54357256]
 [ -9.2640122  -10.59493371  -7.98136707]
 [ -6.6686002   -8.15508292  -8.18001661]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

2.1.2. Convergência do Retorno

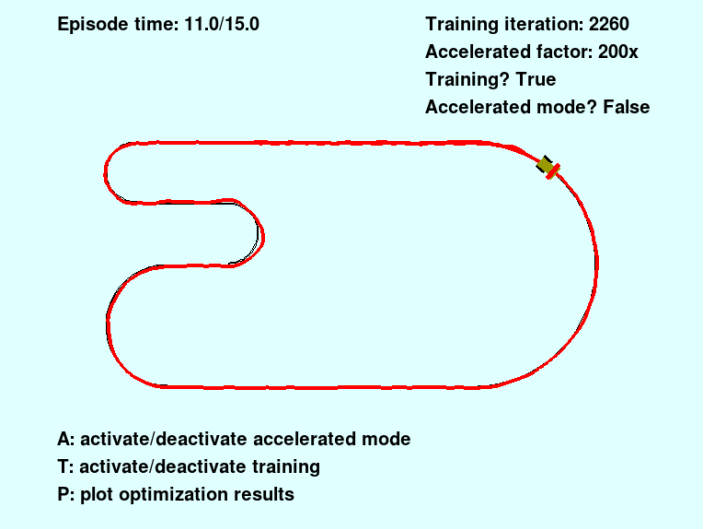


2.1.3. Tabela Q e Política Determinística que Seria Obtida Através de *Greedy*(Q)





2.1.4. Melhor Trajetória Obtida Durante o Aprendizado

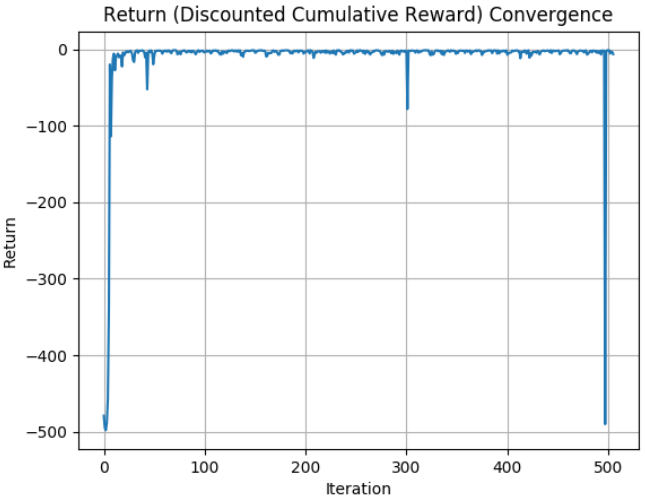


2.2. Q-Learning

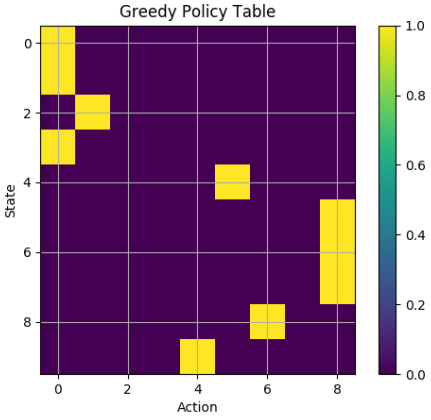
2.2.1. Tabela Ação-Valor e Política *Greedy* Aprendida no Teste com MDP Simples

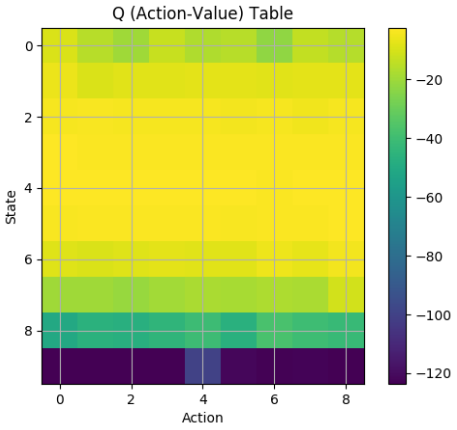
```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.96570291 -1.99       -3.93543179]
 [-3.67897879 -2.9701     -4.03777599]
 [-4.25266327 -3.94039894 -4.16616016]
 [-4.97760341 -4.89611552 -4.8959801 ]
 [-4.33165584 -4.6215259  -3.94039893]
 [-3.62329294 -4.13697079 -2.9701    ]
 [-2.96244449 -3.9318408  -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]
```

2.2.2. Convergência do Retorno

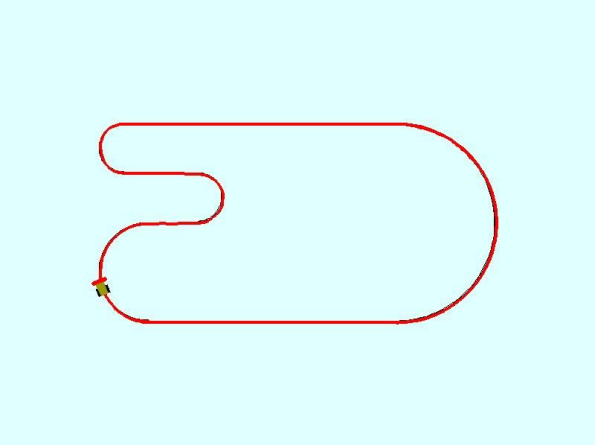


2.2.3. Tabela Q e Política Determinística que Seria Obtida Através de *Greedy*(Q)





2.2.4. Melhor Trajetória Obtida Durante o Aprendizado



3. Discussão dos Resultados

A partir dos resultados pode-se perceber que ambos os algoritmos conseguiram resultados satisfatórios para a resolução do problema do robô seguidor de linha. Com apenas cerca de 500 iterações foi possível obter trajetórias com valores altos, mostrando uma boa política.

Foi possível perceber, entretanto, que o algoritmo de Q-learning obteve resultados um pouco melhores que os observados no SARSA para o mesmo número de iterações, tanto no teste quanto no robô seguidor de linha. Conjectura-se que esses resultados são frutos da característica *off-policy* desse algoritmo, que permite que ele possa aprender também com experiências passadas, acelerando o aprendizado.