Laboratório 11 - Aprendizado por Reforço Livre de Modelo

Marcelo Buga Martins da Silva

CT-213 - Professor Marcos Ricardo Omena de Albuquerque Máximo

04/07/2021

1 Introdução

O propósito desse laboratório foi implementar os algoritmos Sarsa e Q-Learning de aprendizado por reforço livre de modelo para a resolução de um problema de seguidor de linha. Para esse problema, o robô possui velocidade linear constante e deve aprender uma política para sua velocidade angular dado um erro.

Para esse problema, os valores de erro foram discretizados em 9 valores diferentes com a adição de 1 estado para representar linha não-detectada (totalizando 10 estados possíveis) e usa-se 9 valores diferentes para discretizar a ação (igualmente distribuídos entre $-\omega_{max}$ e ω_{max}). Para mapear do estado contínuo para discreto, considera-se o estado discreto mais próximo. A recompensa utilizada foi:

$$reward = -(e/w_I)^2$$

Em que e é o erro e w_I é um fator de normalização associado à largura do sensor de linha do robô. No caso em que o robô não detecta a linha, adotou-se $reward = -5$.

Primeiramente, foram implementados os algoritmos Sarsa e Q-Learning para um problema simples para determinar a melhor ação para se tomar estando em cada casa de um tabuleiro unidimensional de 10 casas,

visando-se chegar a seu extremo na direita. As possíveis ações desse MDP são STOP (ficar parado), LEFT (mover-se para esquerda) e RIGHT (mover-se para direita). Considera-se que ocorre "wrap" nos extremos do corredor: quando se executa LEFT na célula mais à esquerda, o agente surge na célula mais à direita. Analogamente, executar RIGHT na célula mais à direita faz o agente surgir na célula mais à esquerda. O agente recebe recompensa -1 em todas as células, exceto na célula objetivo, em que recebe recompensa 0.

A discussão mais detalhada da implementação dos algoritmos e dos resultados para cada problema são expostos a seguir:

2 Implementação

A implementação dos algoritmos consistiu, primeiro, na implementação de funções *greedy* e *ε-greedy* que foram utilizadas pelo Sarsa e pelo Q-Learning. Essas funções auxiliares são de fácil implementação para um modelo de tabela de ação-valor, bastando retornar a ação de maior valor para um determinado estado para *greedy* e fazer o mesmo com probabilidade $1 - \varepsilon$ para *ε-greedy*, existindo probabilidade ε de o retorno ser uma ação aleatória.

Para o Sarsa, como o algoritmo é *on-policy*, a função *ε-greedy* foi utilizada para aprendizado e execução. Além disso, aplicou-se a fórmula de atualização da tabela de ação-valor, conforme a equação a seguir:

$$Q(S, A) = Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A)\right)$$

Já para o Q-Learning, implementou-se sua *greedy action* e sua lógica de aprendizado, que segue a seguinte equação:

$$Q(S, A) = Q(S, A) + \alpha \left(R + \gamma \max_{a' \in A} Q(S', a') - Q(S, A)\right)$$

3 Caso Teste

Conforme descrito na Introdução, os algoritmos implementados foram testados em um tabuleiro unidimensional a fim de determinar a ação a ser tomada em cada estado do tabuleiro a fim de se chegar na casa mais a direita, levando em conta que, estando na primeira casa e indo para a esquerda, chega-se na casa mais a direita. Os resultados do Sarsa são apresentados na Figura 1 e os do Q-Learning são apresentados na Figura 2:

Pode-se perceber que as políticas aprendidas são coerentes com o esperado. A diferença entre as duas ocorre apenas na melhor ação a se tomar na quinta casa, porém nota-se que, para essa casa em específico, ir para

```
Action-value Table:
[[ -9.18709242  -8.09269062 -10.28971392]
 [-10.42163806  -9.17413678 -11.39938073]
 [-11.24561974 -10.26912747 -11.75671333]
 [-11.78132966 -11.34571987 -12.15119437]
 [-12.40288233 -12.27317196 -12.15973585]
 [-11.52910431 -12.20927876 -11.19233512]
 [-11.09386672 -11.29914749 -10.3390626 ]
 [-10.42717815 -11.19697401  -9.24856538]
 [ -9.50508766 -10.25678783  -8.19164726]
 [ -6.65314889  -8.27037263  -8.10262641]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

Figura 1: Teste do Sarsa para o tabuleiro unidimensional

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.96815809 -1.99       -3.91528402]
 [-3.67663525 -2.9701     -4.42848685]
 [-4.44288511 -3.94039891 -4.32556352]
 [-5.05873229 -4.89558459 -4.89601877]
 [-4.20688373 -4.72186246 -3.94039817]
 [-3.61135159 -4.0458975  -2.9701    ]
 [-2.96139877 -3.93844722 -1.99      ]
 [-1.99       -2.9701     -1.        ]
 [ 0.         -0.99       -0.99      ]]
Greedy policy learnt:
[L, L, L, L, L, R, R, R, R, S]
```

Figura 2: Teste do Q-Learning para o tabuleiro unidimensional

a esquerda ou para a direita é equivalente. Em geral, os valores associados ao Sarsa são menores que os do Q-Learning, o que era esperado pois o Sarsa leva em conta que executa política ε-*greedy*, o que leva a seus valores serem mais punitivos. Observa-se que, no caso do Q-Learning, os módulos dos valores das ações ótimas são muio próximos do número de passos a serem dados naquela direção para chegar ao objetivo.

4 Aprendizado da política do robô seguidor de linha

Foram executados ambos os algoritmos para o aprendizado da política de um robô seguidor de linha, conforme explicada na Introdução. Os resultados para cada um deles, após mais de 500 iterações de treinamento, são mostrados a seguir:

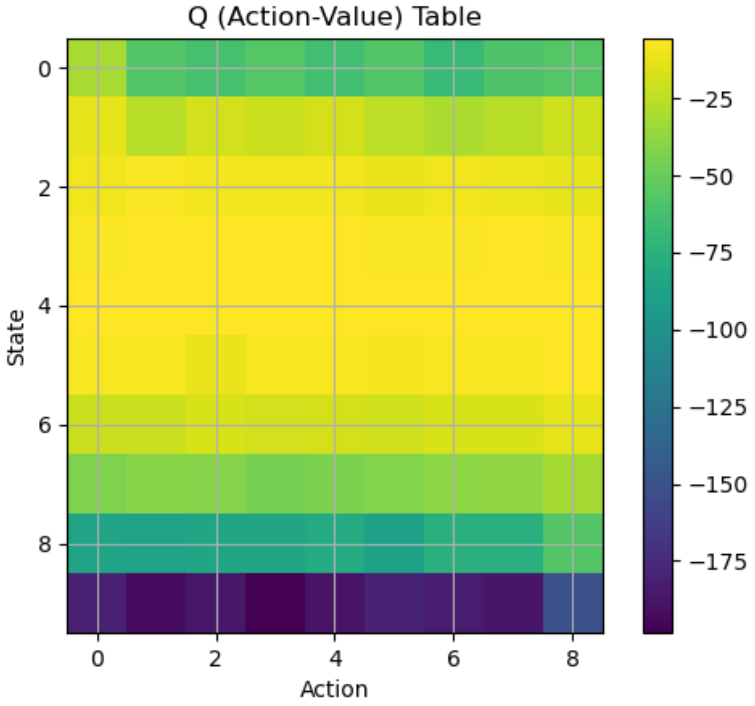

Figura 3: Tabela de ação-valor Sarsa

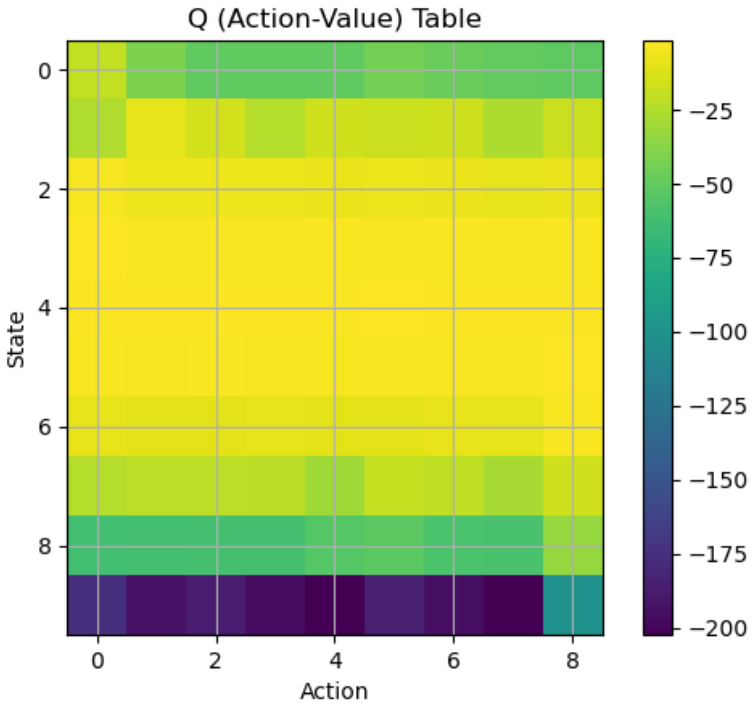

Figura 4: Tabela de ação-valor Q-Learning

Figura 5: Tabela de *greedy policy* Sarsa

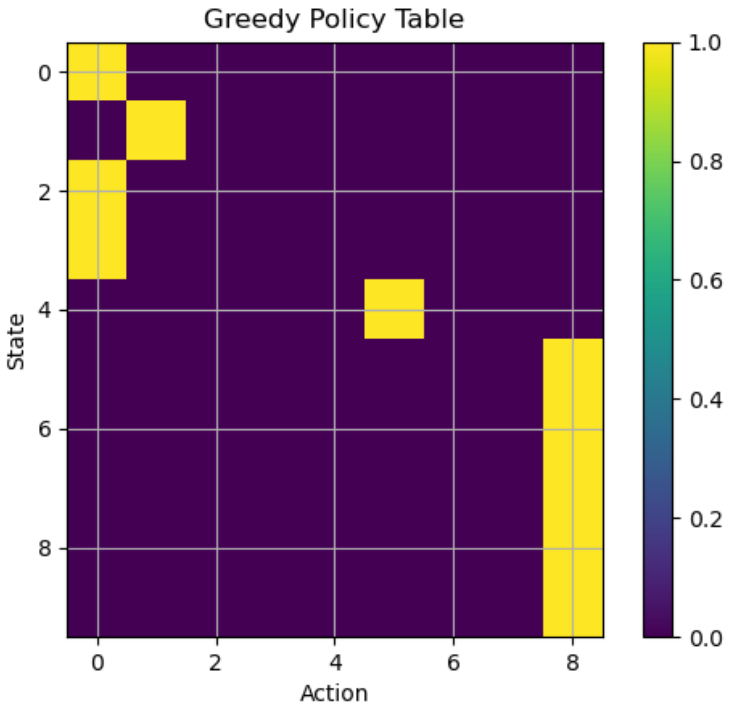

Figura 6: Tabela de *greedy policy* Q-Learning

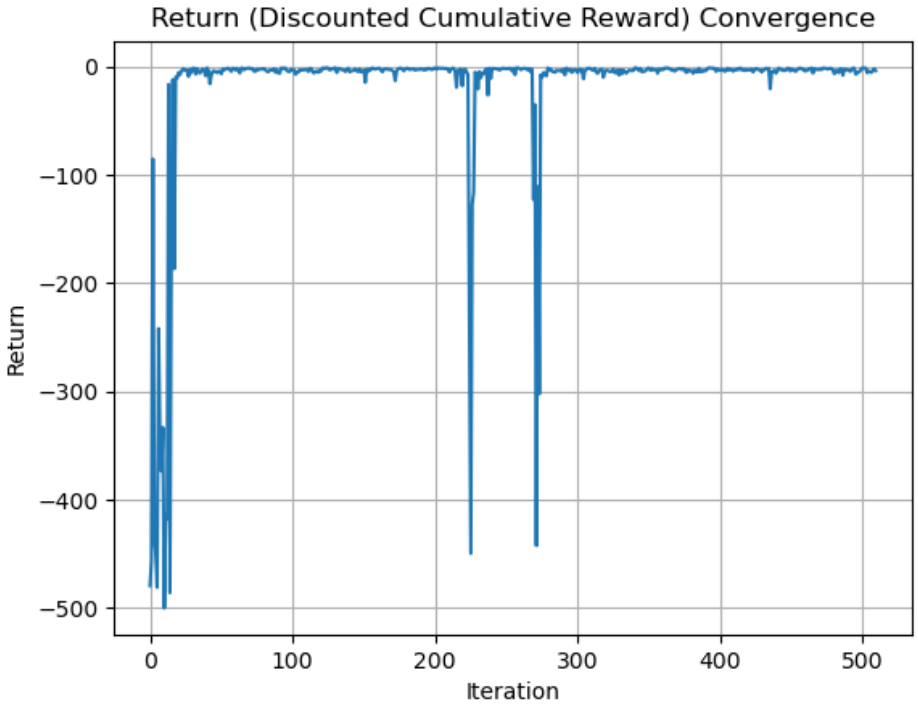

Figura 7: Convergência de retorno para o Sarsa

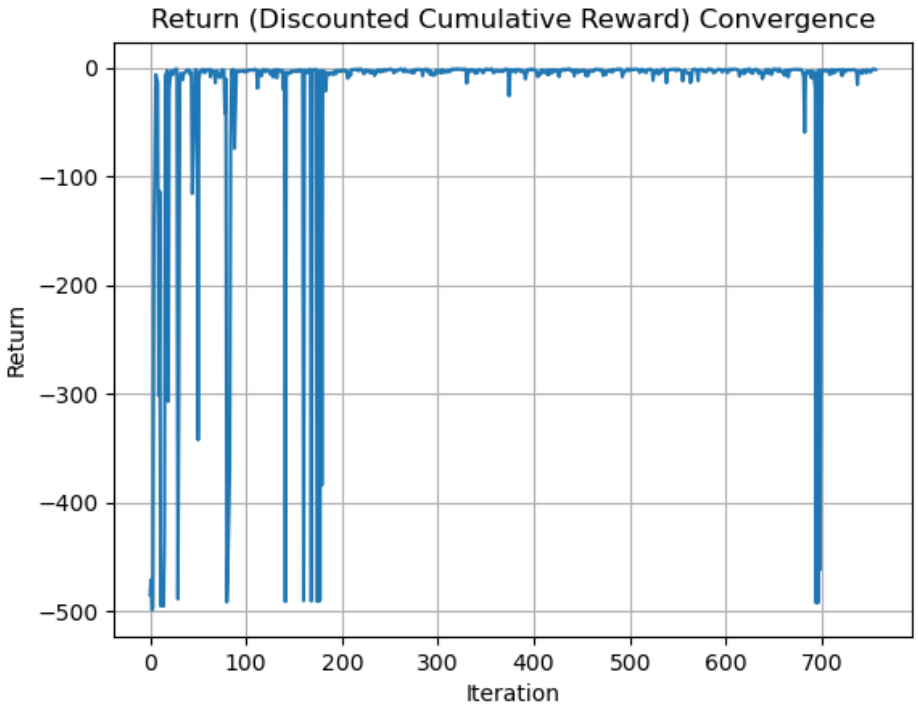

Figura 8: Convergência de retorno para o Q-Learning

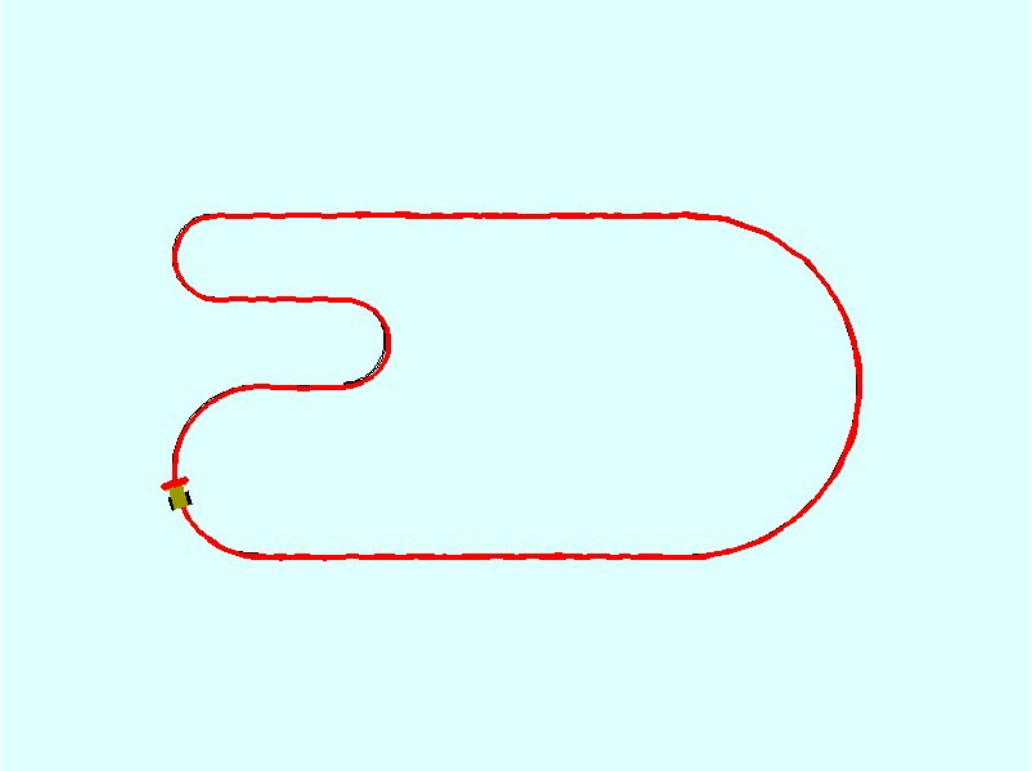

Figura 9: Melhor trajetória para o Sarsa

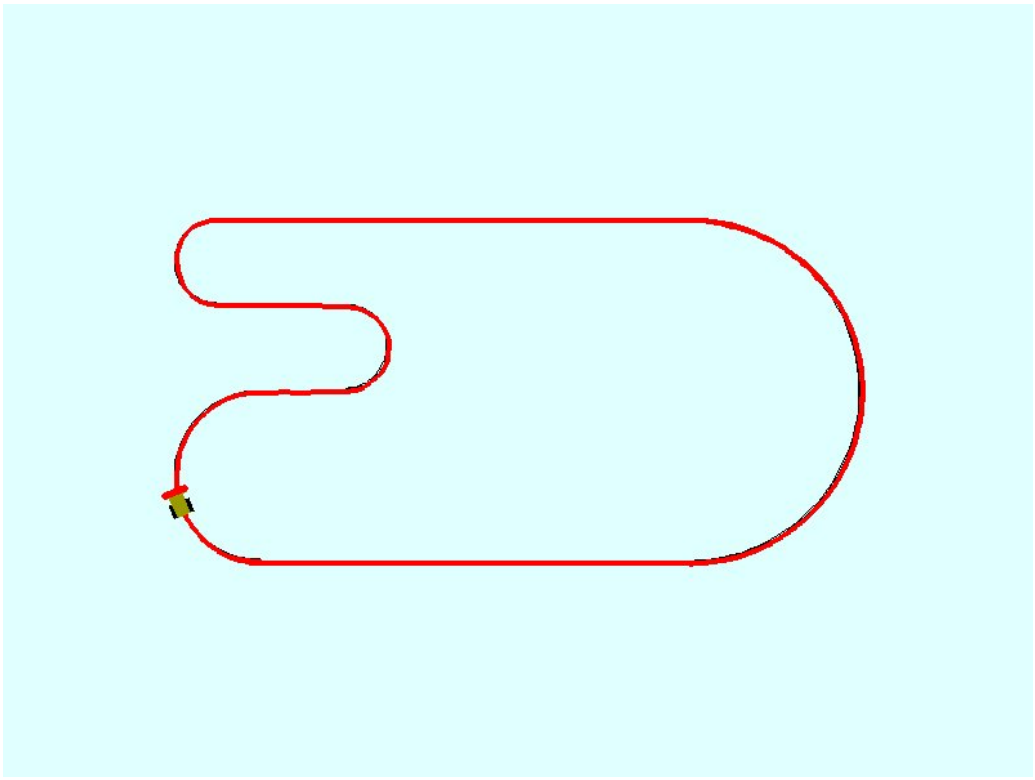

Figura 10: Melhor trajetória para o Q-Learning

Durante o treinamento, foi percebido que a convergência para uma boa trajetória ocorreu de forma bem rápida para os dois algoritmos, de forma que próximo da iteração 20 a maioria das trajetórias já era consideravelmente boa. Durante o treinamento, algumas trajetórias muito erradas foram testadas, mesmo após o algoritmo já ter chegado em uma política muito melhor. Isso é refletido nas Figuras 7 e 8, uma vez que pequenas mudanças na velocidade angular podem causar a perda da linha pelo robô e uma queda muito grande no valor associado a tal política. Não foram percebidas grandes diferenças visuais entre os dois treinamentos. Embora os algoritmos não tenham chegado exatamente nas mesmas políticas, elas foram muito parecidas, o que reflete a semelhança nas melhores trajetórias (Figuras 9 e 10). Além disso, ambas foram muito precisas, saindo muito pouco do circuito, o que confirma a boa implementação dos algoritmos e sua utilidade em problemas de robótica móvel.