Relatório do Laboratório 12: Aprendizado por Reforço Livre de Modelo

Isabelle Ferreira de Oliveira
CT-213 - Engenharia da Computação 2020
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos, Brasil
isabelle.ferreira3000@gmail.com

***Resumo*—Esse relatório documenta a implementação de algoritmos de Aprendizado por Reforço (RL) Livre de Modelo, a saber Sarsa e Q-Learning, e utilizá-los para resolver o problema do robô seguidor de linha.**

***Index Terms*—Aprendizado por reforço, Sarsa, Q-Learning**

I. Implementação

A. Implementação dos algoritmos de RL

1) Função epsilon_greedy_action: A política epsilon-greedy foi implementada da seguinte maneira: gerou-se um número aleatório entre 0 e 1 e, caso esse valor aleatório seja menor que epsilon, então uma ação aleatória é escolhida; caso contrário, é escolhida a ação gulosa, através da chamada de greedy_action.

2) Função greedy_action: Conforme sugerido na seção Dicas do roteiro [1], foi pegue o índice do máximo elemento do array q[state], que é a tabela action-value para o estado naquele momento, e foi retornado esse valor.

3) Função get_greedy_action para algoritmo Sarsa: Retorna a função epsilon_greedy_action, aplicada na tabela action-value q, no estado em questão e com o episolon especificado.

4) Função learn para algoritmo Sarsa: O aprendizado é feito atualizando o valor da tabela de action-value, acrescentando ao valor anterior o resultado de $\alpha * (recompensa + \gamma * q[nextstate][nextaction] - q[state][action])$.

5) Função get_greedy_action para algoritmo Q-Learning: Retorna a função greedy_action, aplicada na tabela action-value q e no estado em questão.

6) Função learn para algoritmo Q-learning: Foi feito de forma análoga ao descrito em Função learn para algortimo Sarsa, I-A4.

B. Aprendizado da política do robô seguidor de linha

Para realizar o aprendizado do robô seguidor de linha, utilizando as duas técnicas implementadas anteriormente (Sarsa e Q-Learning), foi executado o script main.py, alterando-se o valor da variável rl_algorithm, entre os construtores: Sarsa e QLearning, com seus respectivos parâmetros.

II. Resultados e Conclusões

A. Implementação dos algoritmos de RL

Conforme apresentado nas Figuras 1 e 2, pode-se observar resultados equivalentes para ambas as técnicas. Assim, tanto a tabela de action-value possuiu valores similares, como as sequências de ações foram idênticas nas duas situações.

```
Action-value Table:
[[ -9.53107693  -8.69675493 -10.62190887]
 [-10.46744504  -9.59006176 -11.34641517]
 [-10.77957993 -10.37461444 -11.56450722]
 [-11.57970491 -11.24878368 -11.9346405 ]
 [-12.47179686 -12.26901561 -12.20556521]
 [-11.7594528  -12.06217742 -11.18861443]
 [-11.10873604 -11.70958923 -10.20067221]
 [-10.38286681 -11.41848813  -9.45892669]
 [ -9.51028398 -10.28991342  -8.610949  ]
 [ -7.24293802  -8.6162553   -8.59414415]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

Figura 1. Resultado obtido para algoritmo Sarsa, para tabela action-value e a sequência de ações.

```
Action-value Table:
[[ -9.60843699  -8.71243435 -10.4095549 ]
 [-10.60663143  -9.53906908 -11.35042826]
 [-11.06152178 -10.3860185  -11.45310935]
 [-11.74309671 -11.33058737 -12.2666115 ]
 [-12.3233737  -12.24619843 -12.28853041]
 [-11.7258603  -12.31768249 -11.50084476]
 [-11.00920254 -11.48264535 -10.66590364]
 [-10.46664604 -11.57452687  -9.57067003]
 [ -9.37033003 -10.56389603  -8.803751  ]
 [ -7.6143359   -8.70154746  -8.74207702]]
Greedy policy learnt:
[L, L, L, L, L, R, R, R, R, S]
```

Figura 2. Resultado obtido para algoritmo Q-Learning, para tabela action-value e a sequência de ações.

B. Aprendizado da política do robô seguidor de linha

Os resultados do aprendizado da política do robô seguidor de linha, utilizando os algoritmos de Sarsa e Q-Learning estão representados nas Figuras de 3 a 6 e 7 a 10, respectivamente.

bçabjfjfça

1) CORRECT_ACTION_PROB = 0.8
2) GAMMA = 0.98

Primeiro comparando-se os resultados apresentados nas Figuras **??** e **??**, também é possível notar que eles são idênticos, o que é novamente esperado, uma vez que ambas as técnicas levam a convergência dos valores corretos de *policy* e *value*.

Sobre a Figura **??**, a mesma tendência que no primeiro Grid World é observada, ou seja, o *value* calculado é maior em módulo para estados mais distantes do estado objetivo.

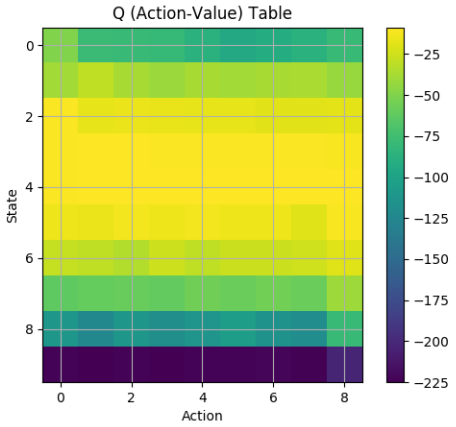

Figura 3. Representação em cores da tabela de action-value calculada, para algoritmo de Sarsa.

.

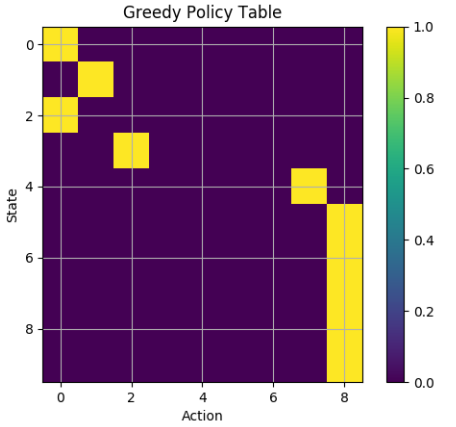

Figura 4. Representação em cores da tabela de greedy-policy calculada, para algoritmo de Sarsa.

.

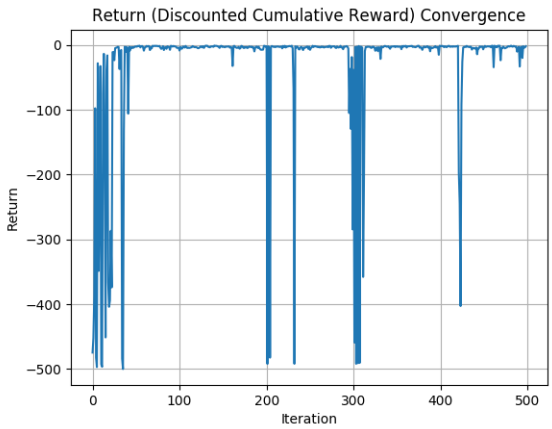

Figura 5. Recompensa acumulada em função das iterações, para algoritmo de Sarsa.

.

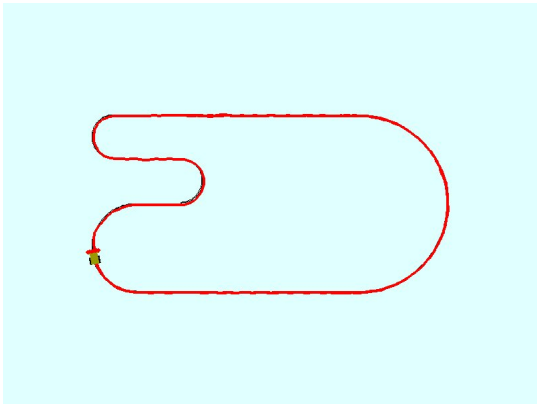

Figura 6. Percurso aprendido pelo algoritmo de Sarsa.

.

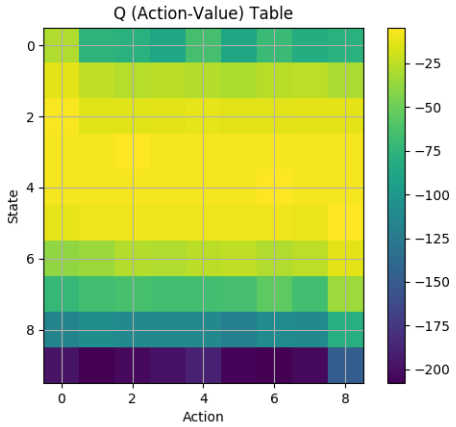

Figura 7. Representação em cores da tabela de action-value calculada, para algoritmo de Q-Learning.

.

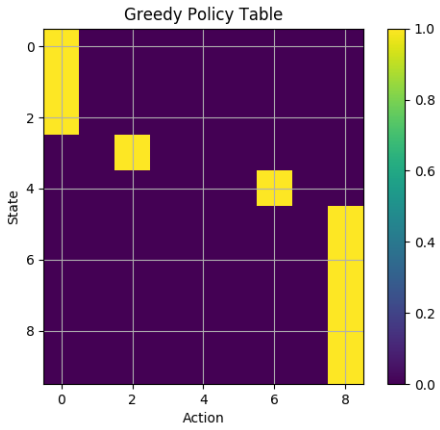

Figura 8. Representação em cores da tabela de greedy-policy calculada, para algoritmo de Q-Learning.

.

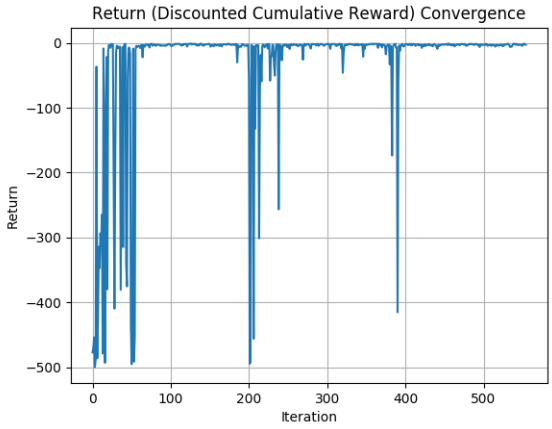

Figura 9. Recompensa acumulada em função das iterações, para algoritmo de Q-Learning.

.

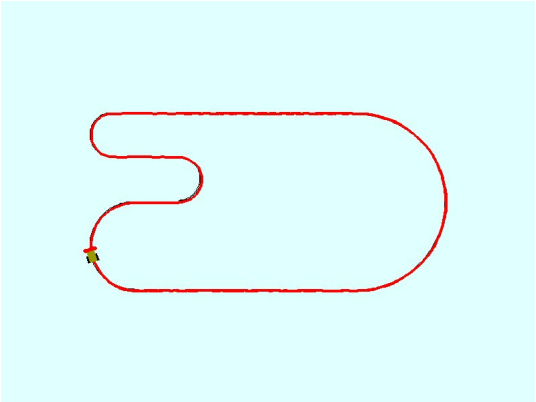

Figura 10. Percurso aprendido pelo algoritmo de Q-Learning.

.

Nos três resultados também é possível notar o *value* 0.0 para o estado objetivo, o que também condiz com o esperado.

Por fim, comparando-se as duas situações de Grid World, é possível notar que, com a adição do desconto *GAMMA*, e agora com a probabilidade de o agente executar uma ação diferente da escolhida para cada estado, tem-se que os *value* referentes a cada estado são maiores em módulo do que os calculados na primeira situação.

Isso se justifica e condiz com o esperado, uma vez que não se sabendo deterministicamente a ação tomada em cada estado, a função valor entende esse estado como "pior"quando comparado a situação na qual *CORRECT_ACTION_PROB* = 1. Além disso, o fator *GAMMA* adiciona mais imediatismo à recompensa das ações do agente, o que também diminui a medida de quão "bom"é determinado estado em comparação a situação no qual todas as recompensas até o objetivo são igualmente contabilizadas.

REFERÊNCIAS

[1] M. Maximo, “Roteiro: Laboratório 11 - Programação Dinâmica”. Instituto Tecnológico de Aeronáutica, Departamento de Computação. CT-213, 2019.