Instituto Tecnológico de Aeronáutica

CT-213: Inteligência Artificial para Robótica Móvel

Lab 11: Aprendizado por Reforço Livre de Modelo

Bruno Benjamim Bertucci - Turma 23.2

1 Introdução

O aprendizado por reforço baseado em modelo, o qual realiza avaliações de política para determinar a política ótima, não pode ser aplicado quando a dinâmica do *Markov Decision Process (MDP)* não é conhecida, como é o caso de várias aplicações mais complexas. Para solucionar esse problema, é necessário usar um método denominado aprendizado por reforço livre de modelo. Algoritmos desse tipo aprendem a partir da experiência adquirida no treinamento. Dentre eles, alguns necessitam que o episódio de treinamento chegue até o fim para realizar o aprendizado, pois utilizam o retorno acumulado no episódio, enquanto outros aprendem a cada passo do episódio. Dois algoritmos importantes, ambos pertencentes a esse segundo grupo, são o *Sarsa* e o *Q-Learning*.

Uma diferença notável entre esses dois algoritos é que o *Sarsa* é *on-policy*, ou seja, ele aprende uma política a partir da experiência gerada seguindo essa mesma política, enquanto o *Q-Learning* é *off-policy*, o que significa que ele aprende a política a partir da experiência adquirida seguindo outra política.

2 Implementação de *Sarsa* e *Q-Learning*

Os algoritmos de aprendizado *Sarsa* e *Q-Learning* foram implementados de forma compatível, para que o uso de ambos seja igual. Para isso, foi criada uma classe inicial chamada *RLAlgorithm*, que armazena uma matriz, inicialmente nula, cujo número de linhas representa o número de estados possíveis, e o número de colunas representa as ações possíveis, bem como parâmetros relevantes para ambos os algoritmos. Além disso, é implementado também um método para realizar ações exploratórias durante o treinamento.

A escolha da ação exploratória é através de uma política denominada ε-greedy, a qual recebe a matriz correspondente à função ação-valor, o estado atual do agente, e um parâmetro ε constante e definido no construtor da classe. Essa política tem probabilidade ε de escolher uma ação aleatória, e probabilidade $1 - \varepsilon$ de esolher uma ação de forma gulosa, ou seja, a ação com maior valor de função ação-valor para o estado considerado. Esse mecanismo permite que o algoritmo não seja excessivamente guloso e explore bem os estados.

Em seguida, foram criadas classes derivadas da *RLAlgorithm*, nas quais foram implementadas as particularidades do *Sarsa* e do *Q-Learning*. Uma dessas particularidades é um método para executar ações após o termino do treinamento, o qual, no caso do *Sarsa*, utiliza a mesma política ε-greedy usada no treinamento, por conta de se tratar de um aprendizado *on-policy*, e no caso do *Q-Learning*, consiste simplesmente na escolha de uma ação de forma gulosa.

Um outro método implementado separadamente para cada algoritmo é aquele que realiza a atualização da função ação-valor, a qual, para o *Sarsa*, é feita de acordo com a Equação 1, e para o *Q-Learning*, é feita conforme a Equação 2, nas quais Q é a função ação-valor, S_t é o estado atual do agente, A_t é a ação que está sendo analisada, S_{t+1} é o estado após a ação A_t ser tomada, e A_{t+1} é a ação considerada após alcançar o estado S_{t+1}, sendo que A_{t+1} só é utilizada no *Sarsa*. Além disso, α é um parâmetro denominado taxa de aprendizado, enquanto γ é um fator de desconto atribuído ao uso de estados futuros no aprendizado.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)) \tag{1}$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R + \gamma \max_{a' \in A} Q(S_{t+1}, a') - Q(S_t, A_t)) \quad (2)$$

2.1 Teste das implementações

Para testar os dois algoritmos implementados, foi criado um problema simples, composto por um corredor de 10 células, ou seja, um vetor unidimensional de 10 estados, e com 3 ações possíveis: permanecer parado, e mover-se à direita ou à esquerda. O objetivo do problema é o agente alcançar a célula mais à direita, por isso, o retorno é de 0 para essa célula e de -1 para as outras. Além disso, quando o agente está na célula mais à esquerda e executa a ação de mover-se à esquerda, ele é levado à célula mais à direita, e vice-versa. Nessa implementação, as ações são determinísticas, ou seja, o agente sempre conseguirá executar a ação e mudar de estado.

Ambos os algoritmos foram utilizados para encontrar a política ótima para o problema descrito. Os resultados, compostos pela matriz correspondente à função ação-valor e pela política encontrada, encontram-se na Figura 1. Para os dois algoritmos, foram utilizados os parâmetros $\alpha = 0,1$, $\varepsilon = 0,1$, e $\gamma = 0,99$. Ao todo, foram realizados 1000 episódios de treinamento, cada um com 1000 iterações.

```
Action-value Table:
[[ -9.48798386  -8.48419102 -10.41364438]
 [-10.62173984  -9.37852154 -11.39722487]
 [-11.22037618 -10.36199514 -11.01345093]
 [-11.60487274 -11.40791886 -12.40833101]
 [-12.44232071 -12.34214036 -12.35924984]
 [-11.89713873 -11.69265273 -11.48629707]
 [-11.15987035 -11.5959917  -10.48770313]
 [-10.69785006 -11.51541759  -9.59473521]
 [ -9.50254842 -10.44030514  -8.41992058]
 [ -7.17239644  -8.39927782  -8.4346507 ]]
Greedy policy learnt:
[L, L, L, L, L, R, R, R, R, S]
```

(a) Algoritmo *Sarsa*.

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.96446506 -1.99       -3.91983613]
 [-3.68217093 -2.9701     -3.99099083]
 [-4.19339351 -3.94039896 -4.70080683]
 [-5.25242075 -4.89423373 -4.89377069]
 [-4.35261216 -4.45811855 -3.94039875]
 [-3.53904371 -3.69703765 -2.9701    ]
 [-2.95535921 -3.91771948 -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]
```

(b) Algoritmo *Q-Learning*.

Figura 1: Teste dos algoritmos de *RL* livres de modelo.

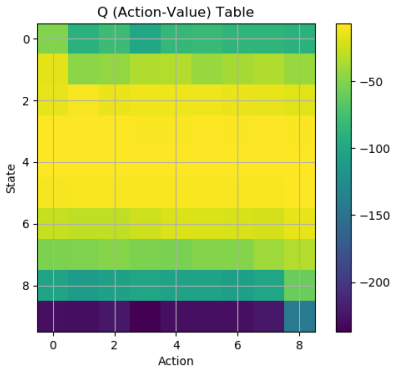
Observou-se que ambos encontraram a política ótima para esse problema. A ação escolhida para a célula equidistante do objetivo, tanto pela esquerda quando pela direita, pode variar entre mover-se à esquerda e à direita para diferentes execuções de ambos os algoritmos, uma vez que, na prática, ambas as ações são plausíveis para uma política ótima.

3 Aplicação em robô seguidor de linha

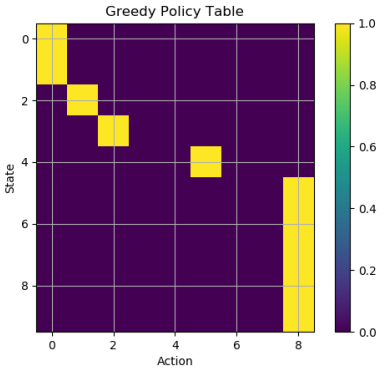
A aplicação escolhida para testar os algoritmos de aprendizado por reforço livres de modelo foi uma simulação da otimização de um robô seguidor de linha, o qual possui uma série de sensores alinhados que permite que ele detecte sua posição com relação a uma linha no solo. Nesse teste, os estados são compostos pelas detecções dos sensores, através dos quais é possível determinar o desvio do agente com relação à linha. Além disso, as ações correspondem a diferentes intensidades para o comando de velocidade angular do agente, em ambos os sentidos de rotação. O objetivo é achar uma política ótima, escolhendo uma velocidade angular ideal para cada estado baseado no desvio do agente, de tal forma que ele consiga seguir a linha da melhor forma possível.

O erro na posição do agente com relação à linha, conforme dado pelos sensores, foi discretizado de tal forma a obter 10 estados possíveis, e os possíveis valores de velocidade angular foram discretizados em 9 ações. Para o teste de ambos os algoritmos de aprendizado por reforço, foi utilizado uma taxa de aprendizado $\alpha = 0,1$, um fator de desconto $\gamma = 0,99$, e $\varepsilon = 0,05$. Dessa forma, a simulação foi executada por pouco mais de 500 iterações em cada teste, sendo que cada iteração consiste de 15 segundos de simulação do robô seguidor de linha.

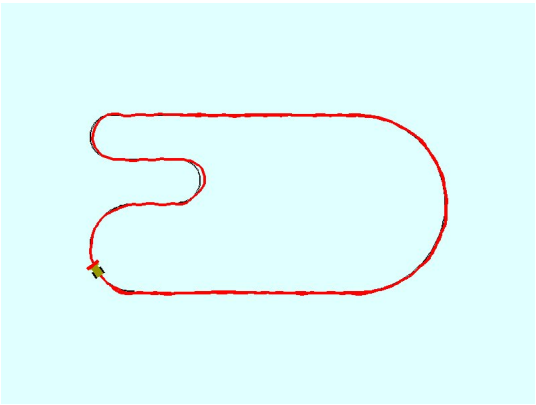
Os gráficos correspondentes às funções ação-valor, às políticas encontradas, aos gráficos de evolução do retorno acumulado de cada iteração, e à trajetória seguida pelo agente com a política otimizada encontram-se na Figura 2 para o *Sarsa* e na Figura 3 para o *Q-Learning*.



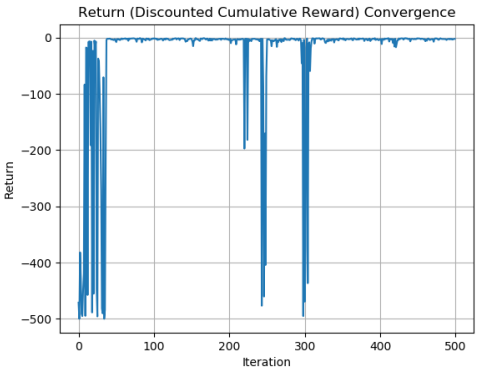
(a) Função ação-valor.



(b) Política encontrada.



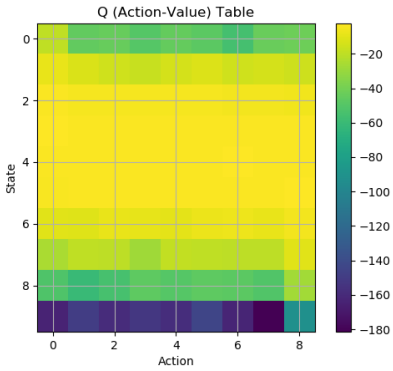
(c) Trajetória do agente.



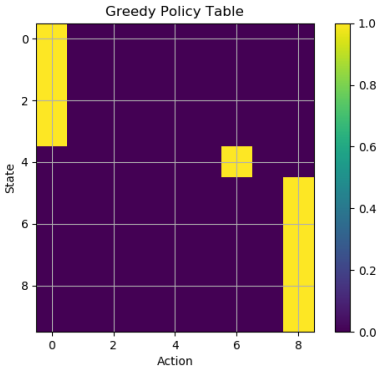
(d) Convergência da recompensa.

Figura 2: Algoritmo *Sarsa*.

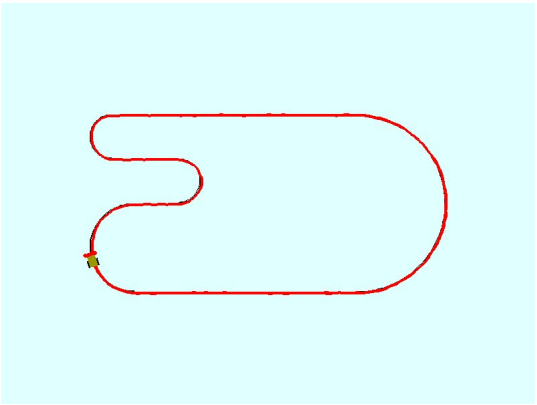
Comparando os dois algoritmos, observou-se que ambos encontraram políticas, em geral, satisfatórias. Pode-se observar, também, que o *Q-Learning* encontrou uma política na qual o agente é capaz de seguir a linha quase perfeitamente, enquanto a política encontrada pelo *Sarsa* apresentou erros ligeiramente maiores, devido ao fato de ele utilizar política ε-greedy na execução.



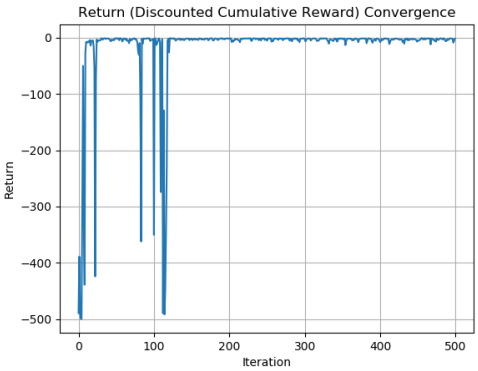
(a) Função ação-valor.



(b) Política encontrada.



(c) Trajetória do agente.



(d) Convergência da recompensa.

Figura 3: Algoritmo *Q-Learning*.

Finalmente, foi possível observar, pelos gráficos de convergência da recompensa acumulada de cada iteração de treinamento, que o *Q-Learning* convergiu mais rápido para uma política próxima da ótima, se comparado ao gráfico do *Sarsa*.