Instituto Tecnológico de Aeronáutica

CT-213

Laboratório 11 - Aprendizado por Reforço Livre de Modelo

Aluno: Pedro Elardenberg Sousa e Souza
Professor: Marcos Ricardo Omena de Albuquerque Maximo

21 de maio de 2024

Conteúdo

1 Resumo

Neste Laboratório, foram implementados algoritmos de Aprendizado por Reforço (RL) Livre de Modelo, a saber *Sarsa* e *Q-Learning*, e com eles resolvido o problema de um robô seguidor de linha.

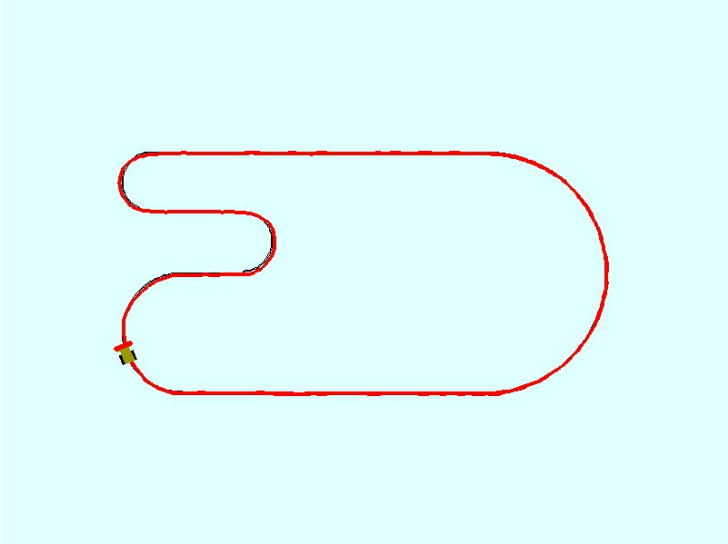

Figura 1: Trajetória do robô seguidor de linha após aprendizado com *Q-Learning*

O objetivo desses algoritmos neste contexto foi de avaliar métodos de aprendizado *on-policy* e *off-policy* no contexto do RL Livre de Modelo. Os algoritmos desenvolvidos resolvem o problema de Aprendizado por Reforço *(Reinforcement Learning - RL)* no caso em que o MDP é conhecido.

Os resultados foram então mostrados e discutidos neste relatório.

2 Introdução

2.1 Aprendizado *On* e *Off-Policy*

No contexto de algoritmos de aprendizado por reforço, um agente *on-policy* aprende o valor com base em sua ação atual **a** derivada da política π atual, enquanto sua contraparte *off-policy* o aprende com base na ação **a*** obtida de uma outra política μ [1]. No *Sarsa*, a avaliação de política é ϵ-greedy. No *Q-learning*, essa política π é a política gananciosa $\pi(s) = greedy(V_k(s))$ em relação ao estado s e à função valor $V_k(s)$[2].

2.2 Política ϵ-greedy

A ideia dessa abordagem é garantir exploração contínua no resultado, pois a ação tomada é a melhor até então (ação gulosa) com uma probabilidade de $1 - \epsilon$. Todas as demais ações terão igual probabilidade:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon, & a = \underset{a' \in A}{argmax}\ Q(s', a') \\ \frac{\epsilon}{m}, & \text{caso contrário} \end{cases} \tag{1}$$

2.3 Sarsa

A ideia do Sarsa é utilizar a recompensa R e o resultado do estado S' após se tomar a ação A' para recalcular o estado S dada a ação A:

$$Q(S, A) = Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A)) \tag{2}$$

Em que α é a taxa de aprendizado e γ é o fator de desconto.

2.4 *Q-Learning*

No *Q-Learning*, segue-se um aprendizado *off-policy*, ou seja, segue política de comportamento μ enquanto aprende política alvo ótima π, em que μ é ϵ-greedy e π é greedy.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma\ \underset{a' \in A}{argmax}\ Q(S_{t+1}, a') - Q(S_t, A_t)) \tag{3}$$

3 Análise dos Resultados

3.1 Implementação dos algoritmos de RL

Após serem implementados os algoritmos Sarsa e *Q-Learning*, os algoritmos foram testados com o código *test_rl.py*. O MDP de teste nesse *script* considera um "corredor" ("tabuleiro" unidimensional) de 10 células. As ações são STOP (ficar parado), LEFT (mover-se para esquerda) e RIGHT (mover-se para direita). As ações são consideradas determinísticas, no sentido de que sempre são executadas com total certeza. Além disso, esse MDP considera 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. Finalmente, o estado objetivo é a célula mais à direita, de modo o agente recebe recompensa -1 em todas as células, exceto na célula objetivo, em que recebe recompensa 0. As figuras 2 e 3 mostram o resultado do teste desses algoritmos.

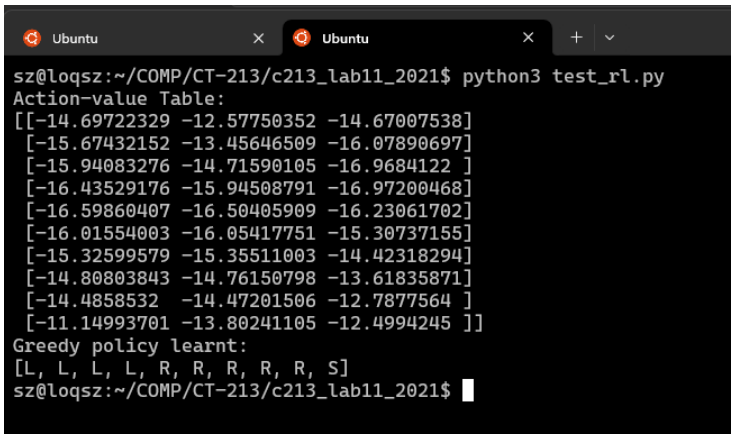

Figura 2: test_rl para o algoritmo Sarsa

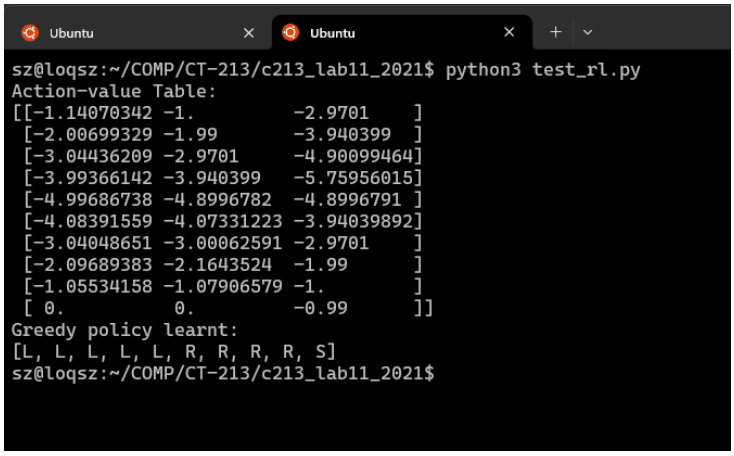

Figura 3: test_rl para o algoritmo *Q-Learning*

Aqui, vê-se que o *Q-Learning* apresenta um resultado mais intuitivo, ou seja, estados a X passos da casa alvo recebem uma recompensa de $-X$. Isso ocorre porque a atualização do estado considera uma política gulosa do estado seguinte, o que previne o algoritmo de fazer muita *exploration*. Em contrapartida, o *Sarsa* possui penalidades maiores para a função valor aprendida, já que a atualização do estado usa uma política ϵ-greedy, que permite mais *exploration*.

3.2 Aprendizado da política do robô seguidor de linha

Após serem testados, os algoritmos foram utilizados para realizar o aprendizado do robô seguidor de linha. Ambos os algoritmos mostraram-se eficientes na tarefa a ser executada, gerando, com menos de 100 iterações, resultados muito próximos aos da figura 1. As figuras 4 e 5 mostram o gráfico de convergência para cada um dos algoritmos.

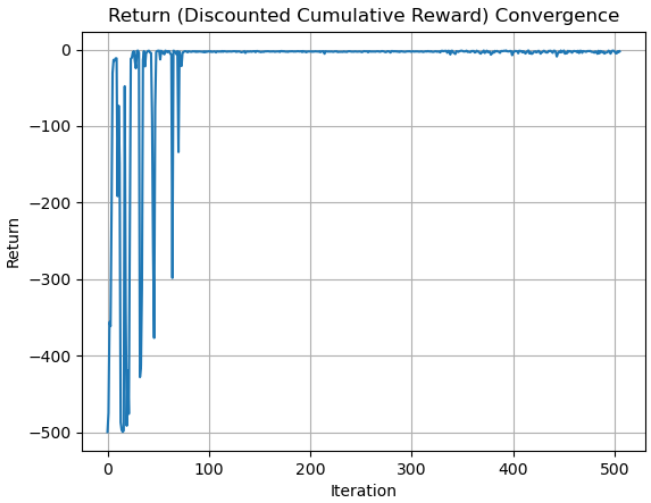

Figura 4: Convergência do algoritmo Sarsa para robô seguidor de linha

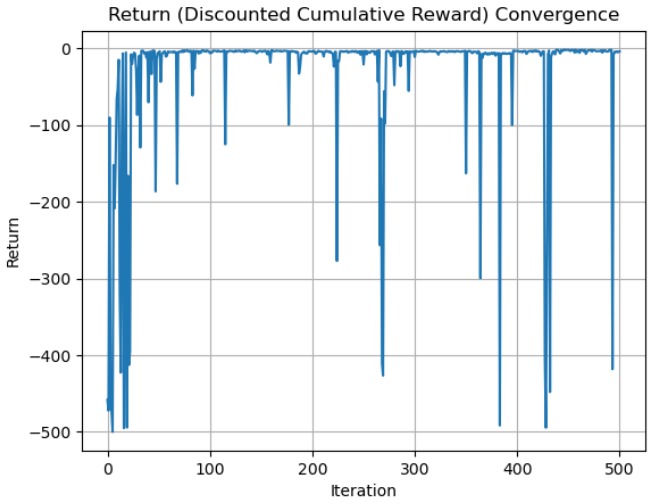

Figura 5: Convergência do algoritmo *Q-Learning* para robô seguidor de linha

Observa-se, pelo caráter de mais *exploration* do Sarsa, que, para este caso, ele apresenta resultados muito ruins (com recompensa na casa dos -400) nas 100 iterações iniciais, mas, no geral, após a convergência, tende a ter resultados mais consistentes, isto é, seguindo a linha mais fielmente. O *Q-Learning*, por sua vez, atinge resultados mais consistentes mais rapidamente (em 20 a 30 iterações), mas ocasionalmente gera resultados ruins.

4 Conclusão

O presente laboratório mostrou que algoritmos de *Reinforcement Learning* livres de modelo são capazes de ser utilizados na resolução de problemas quando não se tem o modelo nem os resultados da função ação-valor desse modelo. Métodos de aprendizado *on* e *off-policy* chegaram em uma solução próxima da ótima, porém, para este caso específico, o algoritmo Sarsa, de aprendizado *on-policy*, mostrou-se mais consistente em seus resultados.

Os algoritmos puderam ser treinados de forma a encontrar a trajetória ótima para o problema dado, com menos de 100 iterações, e chegando a resultados consistentes (recompensa praticamente igual a 0) na maioria das iterações após isso.

Referências

[1] Algoritmo de agente baseado em ia com reinforcement learning. Acessado em 19 de Maio de 2024. URL: `https://www.deeplearningbook.com.br/algoritmo-de-agente-baseado-em-ia-com-reinforcement-learning-parte-2/`.

[2] Marcos Ricardo Omena de Albuquerque Maximo. Ct-213 - aula 12 - aprendizado por reforço livre de modelo. Apostila, 2020. Curso CT-213 - Inteligência Artificial para Robótica Móvel, Instituto Tecnológico de Aeronáutica.