Relatório do Lab 12 de CT-213

Trabalho 12 – Aprendizado de Máquina por Reforço Livre de Modelo

Controlador de um robô *Line Follower* com os algoritmos *Sarsa* e *Q-Learning*

Aluno:

Bruno Costa Alves Freire

Turma:

T 22.4

Professor:

Marcos Ricardo Omena de Albuquerque Máximo

Data:

14/06/2019

Instituto Tecnológico de Aeronáutica – ITA
Departamento de Computação

1. Implementação dos algoritmos de RL (Sarsa e Q-Learning)

A implementação dos algoritmos de *Reinforcement Learning* foi feita no *script* `reinforcement_learning.py`, pode meio de duas classes, `Sarsa` e `QLearning`, ambas derivadas da classe abstrata `RLAlgorithm`. As funcionalidades de ambos os algoritmos residem nos métodos `learn` e `get_greedy_action`, os quais são invocados externamente dentro de um loop para cada iteração de um episódio de treinamento, o qual é feito dentro de um loop sobre todos os episódios de treinamento.

Para o algoritmo Sarsa, que é uma técnica *on-policy*, isto é, aprende a mesma política que utiliza para explorar, a política é escolhida por meio de uma estratégia ε*-greedy* sobre a função de ação-valor. Essa estratégia consiste em escolher, para um dado estado, a ação que maximiza a função ação-valor para aquele estado, com uma probabilidade elevada, mas não determinística, ou seja, uma probabilidade dada por 1 - ε, sendo ε um valor pequeno. Para as demais ações, a probabilidade é distribuída de maneira uniforme. A rigor, a probabilidade da "melhor" ação não é exatamente 1 - ε, uma vez que é somado um termo ε/m, onde m é a quantidade de ações possíveis, de modo a normalizar a distribuição de probabilidade. Dito isso, o método `get_greedy_action` para o Sarsa é simplesmente tomar uma ação ε*-greedy* sobre a função Q.

Para o algoritmo *Q-Learning*, que é *off-policy*, a política exploratória é escolhida de maneira independente da política ótima. O agente executa a política exploratória, mas acumula aprendizado para a política ótima. Enquanto a política exploratória é tomada com uma estratégia ε*-greedy* sobre a função de ação-valor, a política ótima é escolhida de maneira estritamente gulosa ($\varepsilon = 0$).

O método `learn` possui uma leve diferença para cada algoritmo. Em ambos os casos, o que fazemos é atualizar a função ação-valor para um dado estado e uma dada ação, por meio de uma média móvel exponencial, com taxa de aprendizado, α. A priori essa taxa de aprendizado deveria ser escalonada para garantir a convergência dos algoritmos para a política ótima, no entanto, na prática a convergência (ou algo suficientemente convergente) é observada mesmo com α constante.

No algoritmo Sarsa, a média móvel exponencial é realizada entre a função ação-valor no par estado-ação em questão e a estimativa de retorno para aquele par estado-ação, dados o estado e a ação seguintes. Essa estimativa consiste na recompensa acumulada até o momento, acrescida do fator de desconto aplicado sobre a função ação-valor do próximo par de estado e ação.

No algoritmo Q-Learning, por sua vez, a média móvel exponencial é realizada com o Q-retorno esperado para o novo estado, supondo a ação ótima para aquele estado, dessa forma mantendo uma coerência entre o incremento no aprendizado e a política ótima que ele encontra (estratégia gulosa sobre Q).

Após implementados ambos os algoritmos, essa implementação foi testada por meio do *script* `test_rl.py`, por meio de um MDP simples, com um "tabuleiro" unidimensional de 10 casas, onde a última casa à direita é o objetivo. Além disso, o tabuleiro possui estrutura de "reticulado", quer dizer, as extremidades estão conectadas, e em cada casa que não seja o objetivo, a recompensa do MDP é -1. As ações possíveis são mover-se para a direita, mover-se para a esquerda e ficar parado.

O resultado dos testes foi salvo no arquivo `test_output.txt`, e consiste numa representação da função ação-valor ao final do treinamento, e da política ótima encontrada.

Sarsa:

```
Action-value Table:
[[ -9.3838302   -8.35473895 -10.40479917]
 [-10.50241341  -9.23364086 -11.52553145]
 [-11.14525275 -10.5841764  -11.85126005]
 [-11.97894345 -11.55807261 -12.25526843]
 [-12.51358259 -12.39114724 -12.34894396]
 [-11.882916   -12.03409307 -11.61937872]
 [-10.9354176  -11.55701877 -10.70530833]
 [-10.58836115 -11.51036655  -9.32770364]
 [ -9.21934897 -10.72288967  -8.30763947]
 [ -7.36685703  -8.3850545   -8.36258108]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

Na tabela de ação-valor, a primeira coluna corresponde à ação STOP, a segunda à ação LEFT, e a última à ação RIGHT. Cada linha corresponde a um estado, sendo a última linha o estado objetivo.

Q-Learning:

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.96502821 -1.99       -3.92727187]
 [-3.49308985 -2.9701     -4.23546034]
 [-4.40015274 -3.94039867 -4.87071685]
 [-5.13092919 -4.89468389 -4.89492308]
 [-4.38701878 -4.3196495  -3.94039873]
 [-3.67719427 -3.98257351 -2.9701    ]
 [-2.96785871 -3.91416378 -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]
```

É curioso observar que ambos os algoritmos aprenderam políticas ótimas distintas. A rigor, na posição central, não há uma ação ótima.

Contudo, como se tratam de políticas determinísticas (obtidas gulosamente a partir da função ação-valor), não podemos obter uma ação estocástica para o estado central. A razão para termos obtido ações diferentes no Sarsa e no Q-Learning é que o Sarsa constrói a função ação-valor baseado na estimativa do Q-retorno, enquanto o Q-Learning se baseia na otimalidade. Dessa forma, podemos notar que a tabela de ação-valor do Q-Learning é muito mais simétrica que a do Sarsa. Por conta da assimetria da tabela do Sarsa, há uma certa aleatoriedade envolvida que faz com que uma ação seja preferida à outra, arbitrariamente, quando no estado central. Cabe ressaltar ainda que a política obtida pelo Sarsa é sempre ε-*greedy*, não sendo, portanto, exatamente ótima.

2. Treinamento do robô *Line Follower*

Agora que implementamos o aprendizado por reforço livre de modelo, podemos proceder a treinar o robô *Line Follower* em seu nobre objetivo. Diferentemente do Lab4, onde devíamos otimizar os parâmetros do controlador PID do robô, aqui devemos encontrar a política ótima de ação do robô, com base em seu estado. Portanto, precisamos primeiramente definir o espaço de estados. Vamos utilizar o desvio com relação à linha como sendo nosso estado. Como esse valor é contínuo, precisamos discretizar o espaço de estados para que possamos aplicar o Sarsa e o Q-Learning. Analogamente, precisamos definir um espaço de ações discreto.

A estratégia adotada será manter uma velocidade linear constante, invariável, e escolher uma velocidade angular $\omega = \pi(\omega \mid e)$, em função do erro em relação à linha.

Além disso, para que tenhamos um MDP, ainda que não tenhamos o modelo deste, precisamos definir uma função de recompensa. Vamos utilizar como recompensa a função

$$r = \begin{cases} -\left(\frac{e}{w_l}\right)^2, & se\ detectar\ a\ linha \\ -5, & se\ não\ detectar\ a\ linha \end{cases},$$

onde *e* é o erro em relação à linha medido pelo robô, e w_l é um fator de normalização.

Uma vez definidos os parâmetros do nosso MDP livre de modelo, (espaço de estados, espaço de ações, função de recompensa, fator de desconto), estamos prontos para iniciar o treinamento. O robô treinou utilizando cada um dos algoritmos durante 1000 episódios, em cada um dos quais ele tenta executar um circuito dentro de 15 segundos. As figuras de 1 a 8 mostram os resultados do treinamento do robô sob ambos os algoritmos.

Nas figuras de 1 a 4, temos, respectivamente, as estatísticas da função de retorno ao longo do treinamento, a tabela da função ação-valor no final do treinamento, a tabela com a política ótima aprendida, e um *screenshot* da trajetória do robô no circuito seguindo a política ótima, todas para o algoritmo Sarsa.

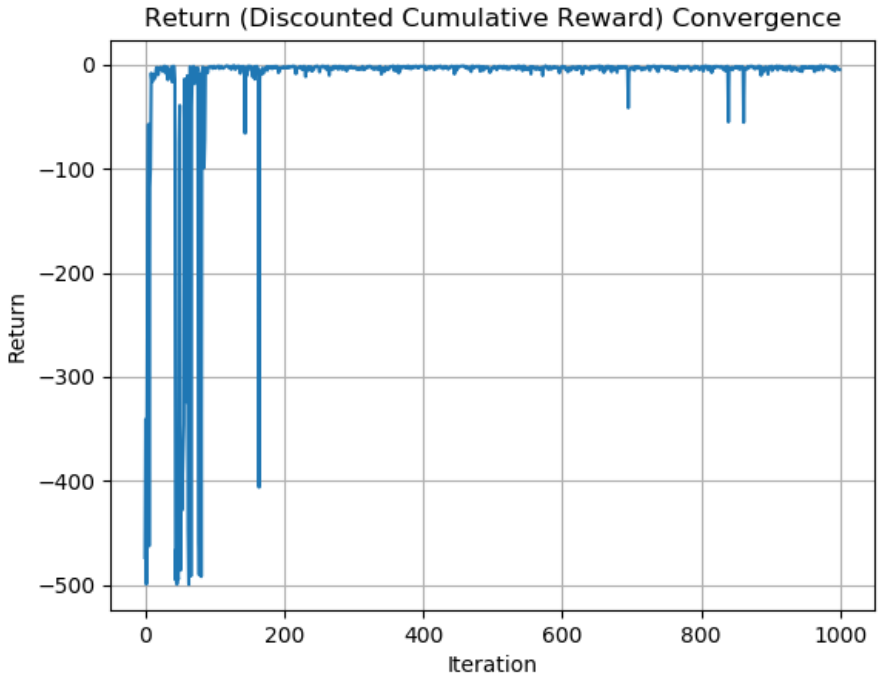

Figura 1: Evolução da função retorno ao longo do treinamento do robô sob o algoritmo Sarsa.

Podemos notar que com 200 iterações o retorno já praticamente converge para 0, apesar de alguns picos negativos mais adiante no treinamento.

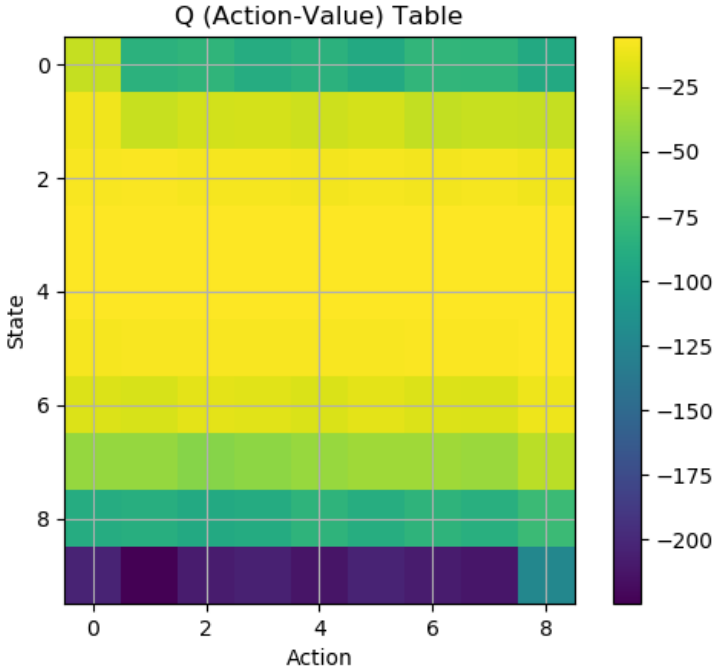

Figura 2: Tabela da função ação-valor aprendida ao longo do treinamento do robô sob o algoritmo Sarsa.

Na figura 2, temos que o último estado representa a situação em que o robô não detecta a linha, por isso o valor de qualquer ação nesse estado é tão baixo. No mais, os estados intermediários são bem avaliados, e os estados com maior detecção de erro apresentam melhor valor para as ações que representam correções mais intensas. Isso mostra que o robô aprendeu a valorizar “ações mais drásticas para situações mais drásticas”.

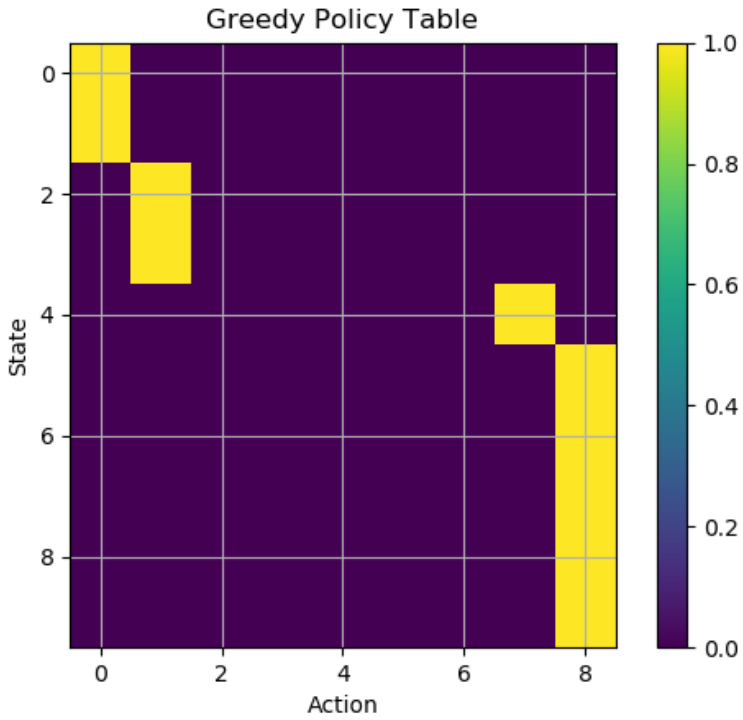

Figura 3: Tabela da política gulosa (ótima) obtida ao final do treinamento pelo algoritmo Sarsa.

Na figura 3 podemos ver pela distribuição de cores, primeiramente, que a política ótima é determinística, como era de se esperar. Ao analisar cada linha, podemos notar que no estado intermediário (4), a ação ótima aprendida não foi a de ficar parado, como era de se esperar. Provavelmente, isso se deve ao fato de que na maioria das vezes em que esse estado era visitado, o robô estava com uma velocidade angular não nula para algum sentido, e portanto era mais vantajoso tomar uma ação corretiva em vez de seguir em linha reta. Além disso, para o estado em que o robô não detecta a linha, a ação aprendida foi de girar o máximo para a direita. Isso pode ser explicado pelo sentido de percurso do circuito, e pela própria geometria do mesmo, onde mais frequentemente o robô precisa fazer curvas para a direita.

Por fim, vejamos na figura 4 um *screenshot* do trajeto executado pelo robô seguindo a política ótima encontrada ao longo do treinamento.

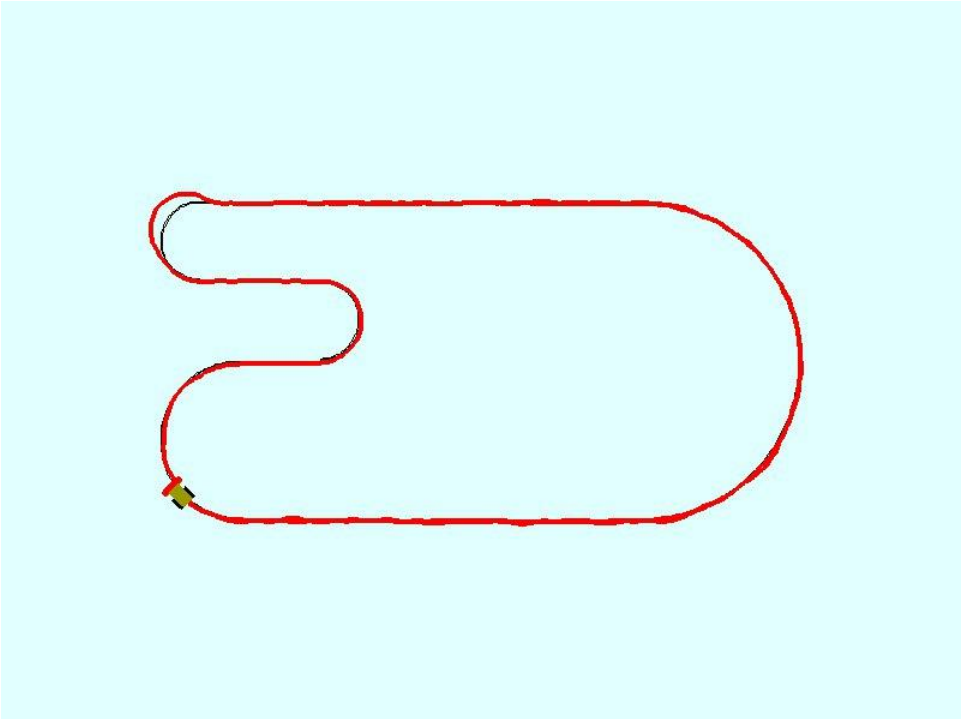

Figura 4: Trajeto executado pela política ótima encontrada pelo algoritmo Sarsa.

Note que o robô quase perde a linha em certo trecho, porém executa o restante do circuito de maneira bastante estável.

A seguir, nas figuras de 5 a 8, temos os resultados para o Q-Learning.

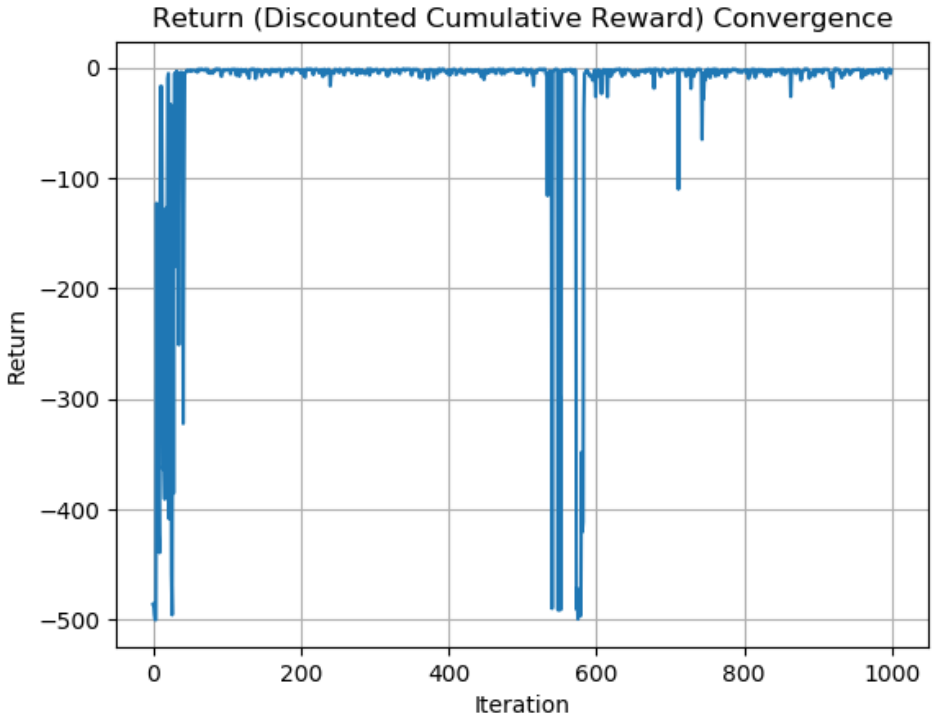

Figura 5: Evolução da função retorno ao longo do treinamento do robô sob o algoritmo Q-Learning.

Pela figura 5 podemos notar que a convergência do retorno para o caso do Q-Learning foi um pouco mais acidentada, havendo alguns picos negativos mesmo após metade do treinamento já ter sido concluído.

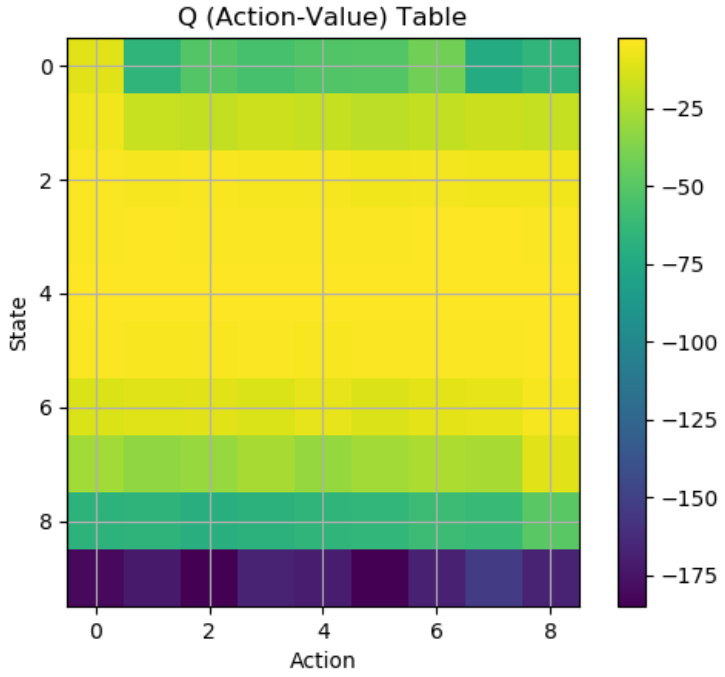

Figura 6: Tabela da função ação-valor aprendida ao longo do treinamento do robô sob o algoritmo Q-Learning.

Na função ação-valor do Q-Learning, temos basicamente o mesmo aspecto que a do Sarsa, com a sutil diferença de que o Q-Learning não deu preferência a nenhuma ação no estado em que o robô não detecta a linha.

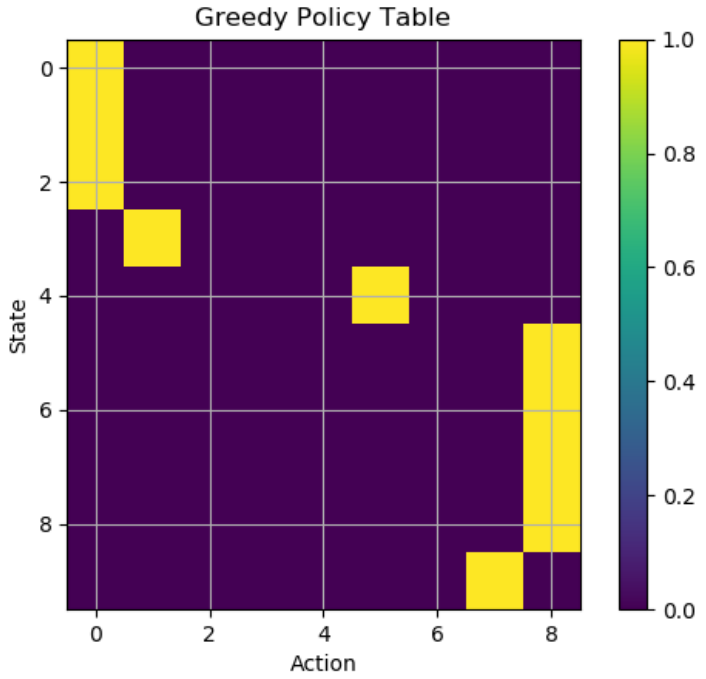

Figura 7: Tabela da política gulosa (ótima) obtida ao final do treinamento pelo algoritmo Q-Learning.

Da figura 7 podemos notar que a política aprendida pelo Q-Learning prevê uma ação mais neutra para o estado central, ao mesmo tempo que não prevê uma ação tão drástica para quando o robô perde a linha. Ainda assim, podemos observar uma certa assimetria na escolhas das ações para os estados, o que novamente pode ser explicado pela geometria e sentido de percurso do circuito.

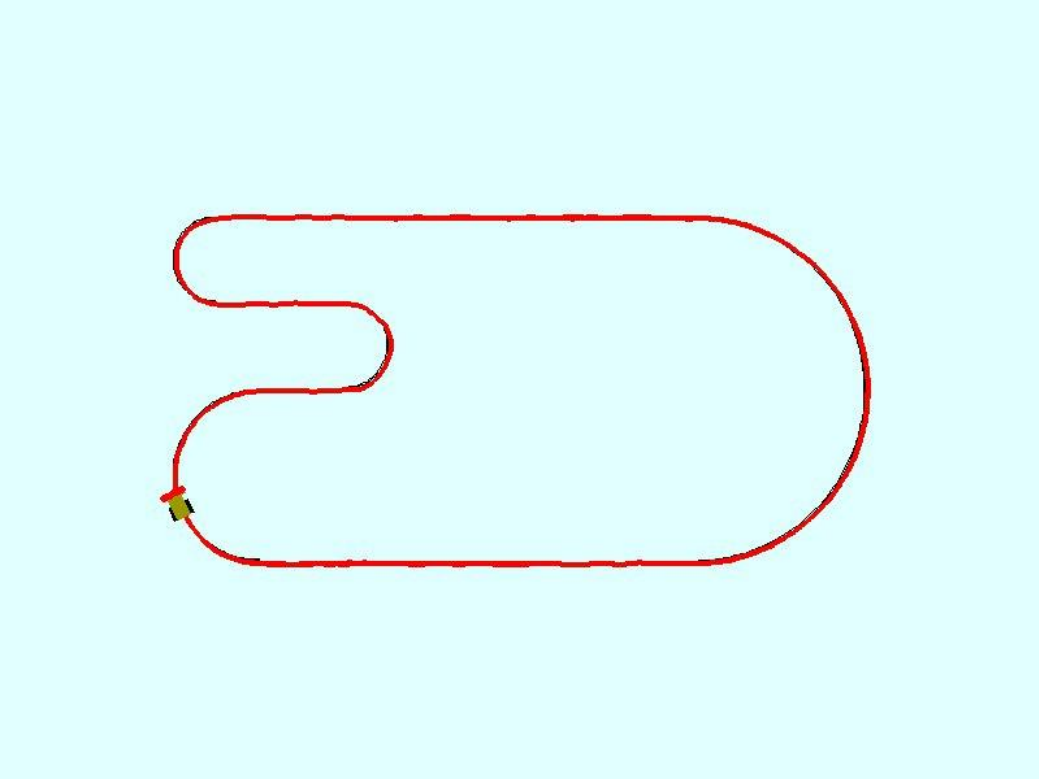

Figura 8: Trajeto executado pela política ótima encontrada pelo algoritmo Q-Learning.

Por fim, vemos que o trajeto desferido pela política ótima do Q-Learning é aparentemente perfeito, sem grandes desvios em nenhum trecho.

Isso nos permite comparar o desempenho dos algoritmos Q-Learning e Sarsa, levando em conta a convergência do retorno ao longo do treinamento e a qualidade do percurso desenvolvido pela política ótima. Sabemos que o Sarsa é um algoritmo *on-policy*, que aprende a política enquanto executa a mesma, e que por necessidade de exploração, essa política é ε*-greedy*. Como não implementamos escalonamento em ε, é de se esperar que essa solução seja um pouco pior que a política ótima. Por outro lado, vimos que ao longo do treinamento do Q-Learning, houve muito mais picos negativos em momentos já avançados do treinamento, o que evidencia um aspecto “temerário” do Q-Learning, que executa políticas exploratórias sem perceber o peso dessa exploração, e que por essa razão, regularmente incorre em políticas perigosas, responsáveis pelos picos negativos da função retorno. O Sarsa, por sua vez, assume uma postura mais cautelosa, embora não alcance a otimalidade, enquanto o Q-Learning atinge a política ótima ao preço de alguns descuidos.