Algoritmos de aprendizado por reforço: uma implementação em Python do Q-learning

Leonardo Loureiro Costa
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, Brazil
leonardo.costa@unifesp.br

***Abstract*—Este relatório aborda a aplicação do algoritmo Q-learning, um método de aprendizado por reforço, para resolver um problema de aprendizado por reforço envolvendo um táxi que deve aprender a buscar e levar um passageiro ao destino. Foram realizados dois experimentos: no primeiro, o táxi escolhe ações aleatoriamente, levando em média 2461 épocas para concluir cada episódio. No segundo experimento, o Q-learning foi aplicado, resultando em uma conclusão muito mais rápida, com uma média de cerca de 13 épocas por episódio. Esses resultados demonstram que o Q-learning teve um impacto significativo no desempenho, tornando o treinamento mais eficiente e rápido. Isso destaca a eficácia do Q-learning na otimização de políticas de ação em ambientes de aprendizado por reforço.**

I. Introdução

A inteligência artificial introduz o paradigma da computação que constroi sistemas e algoritmos capazes de aprenderem, isto é: se tornarem melhores em realizar determinadas tarefas.

A aprendizagem por reforço preenche a lacuna entre os algoritmos de aprendizado supervisionado e não supervisionado. Ela oferece soluções onde agentes aprendem a interagir com um ambiente através de um sistema de recompensas e punições.

O Q-learning, um dos primeiros e mais importantes algoritmo de RL, soluciona o problema de aprendizado por reforço encontrando os valores ideias para uma política ϵ-gulosa de forma a maximizar a soma cumulativa descontada do retorno.

II. Fundamentação teórica

A. Aprendizado por reforço (RL)

Em inteligência artificial, aprender pode ser definido como a capacidade de um sistema ou algoritmo de obter conhecimento e melhorar seu desempenho [7].

O aprendizado de máquina é uma subárea da inteligência artificial que pode ser dividida em três grandes categorias: aprendizado supervisionado, aprendizado não supervisionado e aprendizado semi-supervisionado, também conhecido como aprendizado por reforço [7].

No aprendizado supervisionado, máquinas são treinadas em conjuntos de dados rotulados, isto é: temos uma determinada entrada X, uma saída y, e uma função f que mapeia os elementos da entrada na saída. O objetivo é encontrar uma função $h(x)$ que se aproxime o máximo possível de $f(x)$ [7] - encontrar uma função *hipótese* que se aproxime da função verdadeira.

Por outro lado, o aprendizado não supervisionado lida com dados não rotulados, a máquina não busca encontrar y dado X, e sim buscar identificar padrões, clusters ou relações dentro dos dados que foram providenciados [7].

O aprendizado por reforço, por sua vez, se assemelha ao processo de aprendizado biológico humano, uma vez que permite que máquinas aprendam por meio de interação e exploração. Esse algoritmo é composto por cinco conceitos principais: Agente, Ação, Ambiente, Estados e Recompensa, como ilustrado na Figura 1 [3].

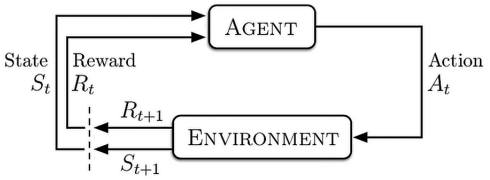

Fig. 1: Diagrama da estrutura de aprendizado por reforço

O agente é a entidade que interage com o ambiente, toma ações e recebe recompensas com base nas decisões tomadas [8].

Um estado pode ser representado como $s \in S$, um elemento abstrato de um conjunto finito de possíveis estados do ambiente [8].

Uma ação pode ser descrita como $a \in A(s)$, um elemento pertencente a um conjunto finito de ações associado a cada estado [8].

Uma recompensa pode ser definida como um elemento de um subconjunto dos números reais - $r \in R \subset \mathbb{R}$ - como, por exemplo, $r \in \{-1, 0, 1\}$ [8].

A interação entre o agente e o ambiente considera que o agente toma uma determinada ação $A_t = a$ e está em um determinado estado $S_t = s$. Podemos definir formalmente essa interação com a equação 1. O próximo estado e a recompensa imediata dependem exclusivamente do estado atual e da ação escolhida. Esse comportamento classifica o aprendizado semi-supervisionado como um Processo Markoviano.

$$p(s', r|s, a) = Prob(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a) \quad (1)$$

A forma como essa interação é realizada, ou seja, como o agente escolhe a ação a ser tomada dado o estado atual, é chamada de política [8]. Esse conceito é definido como $\pi(a|s)$ quando estocástico e $a = \pi(s)$ quando determinístico.

O que determina uma boa política, ou seja, uma boa escolha de ações, é o retorno. O retorno G_t, definido na equação 2, é a soma cumulativa descontada das recompensas obtidas pelo agente durante um episódio. Quanto mais próximo o valor de γ estiver de 0, mais ênfase será dada às recompensas obtidas imediatamente.

$$\begin{aligned} G_t &= \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \\ 0 &\leq \gamma \leq 1 \\ t &\in \{0, 1, \ldots, T\} \end{aligned} \tag{2}$$

O objetivo do aprendizado por reforço é maximizar a soma das recompensas acumuladas ao longo do tempo..

B. Algoritmo Q-learning

Algorithm 1 Algoritmo Q-learning

```
Inicializar a matriz Q com valores 0
Declarar hiperparâmetros: taxa de aprendizado α, taxa de desconto γ, taxa de exploração ϵ
for Quantidade de episódios do
    Inicializar o estado s
    while episódio não terminar do
        Escolher uma ação a usando uma política ϵ-greedy
        Com a ação a, observar a recompensa r e o novo estado s'
        Atualizar Q(s, a) usando a equação 4
        Atualizar o estado atual: s ← s'
    end while
end for
```

Primeiro, construímos uma matriz Q de dimensões $|estados| \times |ações|$ que contém, em cada célula, as recompensas quando uma determinada ação $a \in A(s)$ é tomada no estado s. Valores negativos representam punições, enquanto valores positivos representam recompensas.

Três hiperparâmetros do algoritmo precisam ser declarados: α, a taxa de aprendizado; γ, a taxa de desconto; e ϵ, a taxa de exploração.

Os hiperparâmetros α e γ controlam o equilíbrio entre o valor das recompensas nos estados s_t e s_{t_1}. Por outro lado, o hiperparâmetro ϵ controla o grau de exploração do algoritmo.

Ao determinar a melhor política, é necessário equilibrar dois conceitos importantes: exploração e exploração.

A exploração refere-se à capacidade do agente de explorar novos caminhos, ou seja, não escolher sempre a ação que leva à recompensa ótima, a fim de experimentar diferentes possibilidades. A exploração, por sua vez, envolve uma busca mais gananciosa, selecionando sempre a ação que oferece a recompensa de maior valor [8].

Dessa forma, uma ação a_t pode ser selecionada conforme a equação 3.

$$a_t = \begin{cases} a_t^* \text{ com probabilidade } 1 - \epsilon \text{ (exploração)} \\ \text{ação aleatória com probabilidade } \epsilon \text{ (exploração)} \end{cases} \tag{3}$$

Em seguida, inicializamos a matriz Q, onde para cada par (s, a) temos $Q(s, a) = 0$.

Observamos então um estado inicial aleatório s.

Após isso, escolhemos uma ação usando uma política ϵ-greedy, ou seja, com probabilidade ϵ, escolhemos uma ação aleatória e, com probabilidade $1-\epsilon$, escolhemos a ação ótima. Essa ação resulta em uma recompensa r e um novo estado s'.

Em seguida, atualizamos a matriz $Q(s, a)$ usando a equação 4.

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a')) \tag{4}$$

Por fim, atualizamos o estado $s \leftarrow s'$. Repetimos esse processo, desde a escolha da ação até a atualização do estado s, até que um número limite de episódios seja alcançado ou até que um estado final seja alcançado.

III. Objetivos

O objetivo desse relatório é evidenciar o funcionamento do algoritmo Q-learning e avaliar seu impacto no desempenho de um problema de aprendizado supervisionado comparado com algoritmos que não aprendem nada e apenas tomam ações aleatoriamente.

IV. Metodologia

O algoritmo Q-learning será implementado em Python [1] para solucionar um problema de aprendizado por reforço: um taxi deve aprender qual caminho fazer para buscar e levar um passageiro até o destino. O algoritmo implementado é uma modificação do código poidenciado no webste [4].

A implementação do ambiente é feita por meio da biblioteca gym [6]. Disponibilizada pela OpenAI, essa biblioteca proporcional diversos ambientes para simular aprendizado por reforço.

O ambiente 'Taxi-v3' [5] será selecionado. Esse ambiente possui 500 estados e 6 ações, refentes ao movimento do taxi nas direções norte, sul, leste, oeste e às operações de *pickup* e *dropoff*.

Após a configuração do ambiente dois experimentos serão realizados:

1) Política de escolha aleatória sempre
2) Política ϵ-greedy com Recompensas atualizadas pelo Q-learning

Cada um dos experimentos será executado por 100 episódeos e o objetivo é analisar quantas épocas são necessárias para que cada episódeo seja concluído, em média.

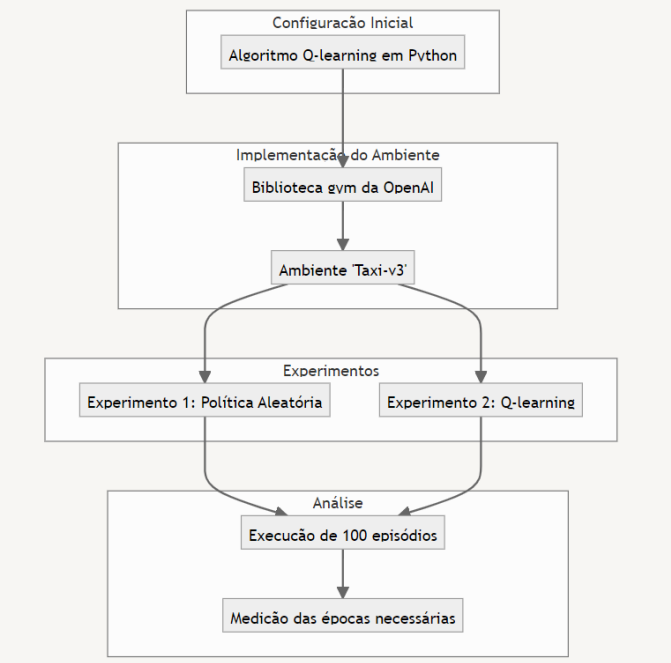

Fig. 2: Diagrama de metodologia dos experimentos

V. Materiais

Os materiais usados para esse experimento foram:

1) O ambiente Jupyter Notebook do Google Colab [2]
2) O ambiente *'Taxi-v3'* [5] disponibilizado na biblioteca *gym* [6] da *OpenAI*

VI. Resultados

A. Experimento 1

O experimento 1 foi realizado por 100 episódeos. Em média foram necessárias 2461 épocas para que o estado final fosse atingido.

Esse expeirmento utiliza como política apenas uma distribuição uniforme das probabilidades de se selecionar uma deteminada ação $a \in A(s)$.

B. Experimento 2

O experimento 2 foi realizado por 100 episódeos também. Em média, foram necessáras apenas cerca de 13 épocas por episódeo para que o estado final fosse atingido.

Comparando os resultados obtidos no experimento 1 e no experimento 2 temos que utilizar o algoritmo Q-learning resulta em alcançar o objetivo do probelam cerca de 190 vezes mais rápido com um treinamento que durou apenas 87 segundos.

VII. Conclusão

Analisando os resultados do experimento 1 e do experimento 2 podemos concluir que o uso do algoritmo Q-learning resultou em um impacto significativo no desempenho do problema presente no ambiente *Taxi-v3*.

O experimento 1 demandou um número maior de épocas, visto que a seleção aleatória de ações não selecionava o caminho ótimo.

O experimento 2 demonstrou um desempenho superior, na medida em que implementava de fato um algoritmo de aprendizado.

Essa diferença de desempenho destaca a eficácia do Q-learning em aprender e otimizar valores de recompensa. O agente que utiliza Q-learning consegue mais eficientemente navegar pelo ambiente, resultando em uma soma descontada maior.

References

[1] Leonardo Loureiro Costa. Algoritmo q-learning em python. https://github.com/Leonardo-Costa/reinforcement$_learning$.
[2] Google LLC. Google colab, 2023. acessado em 26/09/2023.
[3] Mutual Information. Video: Reinforcement learning, by the book, 2022. timestamp 4:42 acessado em 23/09/2023.
[4] Satwik Kansal. Reinforcement q-learning from scratch in python with openai gym. https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/. acessado em 26/09/2023.
[5] OpenAI. Gym. https://www.gymlibrary.dev/environments /toy$_text/taxi/.acessadoem26/09/2023$.
[6] OpenAI. Gym is a standard api for reinforcement learning, and a diverse collection of reference environments. https://www.gymlibrary.dev/. acessado em 26/09/2023.
[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2021. Page 693-696.
[8] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.