Introdução à Inteligência Artificial Q-Learning

Samuel Lipovetsky[1]

[1]Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

1. Introdução

Neste trabalho, foi implementado um algoritmo de aprendizado por reforço conhecido como Q-learning. O Q-learning é uma técnica amplamente utilizada para ensinar um agente a tomar decisões em um ambiente desconhecido, com base em recompensas recebidas por suas ações.Para isso, aplicamos o Q-learning a um problema específico, onde nosso agente precisa navegar em um grid, representando um ambiente com obstáculos e objetivos. O objetivo do agente é encontrar a melhor rota para alcançar um objetivo específico, evitando obstáculos e maximizando a recompensa acumulada ao longo do caminho. Foi implementado as funções necessárias para o funcionamento do algoritmo, como a definição das ações possíveis em cada estado, a inicialização da Q-table, a escolha da ação ótima, o cálculo das recompensas e a atualização da Q-table com base nas interações com o ambiente.

Além disso, foi utilizada visualizações gráficas para acompanhar o progresso do agente, como a representação do grid com os Q-value e ,animação da trajetória percorrida pelo agente durante o treinamento e diversos gráficos que demonstra o avanço do treinamento do agente de acordo com o número de passos.

2. Funções Implementadas

A implementação das funções descritas a seguir é fundamental para o correto funcionamento do algoritmo de Q-learning. Cada uma delas desempenha um papel específico e crucial no processo de aprendizado do agente.

`get_available_actions(grid, state)`: Retorna uma lista de ações disponíveis a partir de um determinado estado em um grid. Verifica se é possível mover para cima, para a direita, para baixo ou para a esquerda a partir do estado atual.

`initialize_q_table(tamanho_grid, num_acoes)`: Inicializa a Q-table com zeros. A Q-table é uma matriz tridimensional que representa o valor de ação para cada estado e ação possíveis.

`choose_optimal_action(q_table, state, grid)`: Escolhe a ação ótima com base na Q-table para um determinado estado. Há também uma probabilidade de 20% de escolher uma ação aleatória para fins de exploração.

`get_reward(grid, state, default_reward)`: Retorna a recompensa para um determinado estado no grid. A recompensa pode ser 1, -1, o valor padrão ou `None`, dependendo do valor da célula no grid.

`read_input_file(file_path)`: Lê um arquivo de entrada que contém as configurações para a execução do algoritmo Q-Learning. Retorna os valores de iterações, taxa de aprendizado, fator de desconto, recompensa padrão, taxa de exploração, tamanho do grid e o próprio grid.

`get_next_state(state, action)`: Retorna o próximo estado com base no estado atual e na ação escolhida.

`q_learning(grid, num_episodes, learning_rate, discount_factor, exploration_rate, default_reward, start)`: Implementa o algoritmo Q-Learning. O agente realiza iterações no ambiente, escolhe ações com base em uma política de exploração e atualiza a Q-table com base nas recompensas obtidas.

`find_agent(matrix)`: Encontra a posição do agente no grid.

`get_max_elements(matrix, grid)`: Obtém o maior elemento da Q-table para cada posição no grid. Retorna uma matriz com os maiores valores de Q e uma matriz de índices que representa a melhor ação em cada posição.

3. Explorando os resultados

Com o intuito de apresentar graficamente os resultados , foram feitos gráficos que discutem algumas métricas evoluindo com o número de episódios e ainda gráficos que comparam hyper-parâmetros do algoritmo.

3.1. O exemplo utilizado

Foi utilizado um grid 6x6 , demonstrado na figura 1, em que o agente tem o label A ,os blocos em roxo escuro são obstáculos , D é um estado de derrota e V um estado vitorioso.

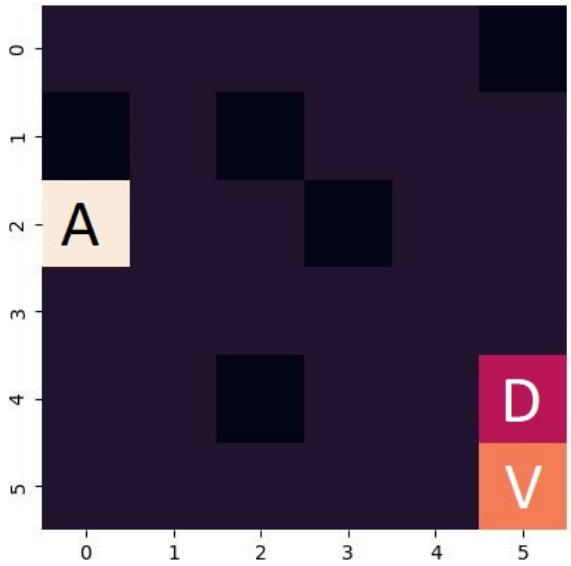


Figure 1. O espaço utilizado

Além disso foram utilizados 5000 passos, alpha = 0.1, fator de desconto = 0.8, recompensa padrao = -0.06 e epsilon = 0.1

4. Número de passos por episódios

O gráfico a seguir demonstra que o algoritmo converge para um número de passos após N iterações e que a convergência nunca chega a ser uma linha reta devido a incerteza no movimento do agente e da estrategia usando o epsilon-greedy.

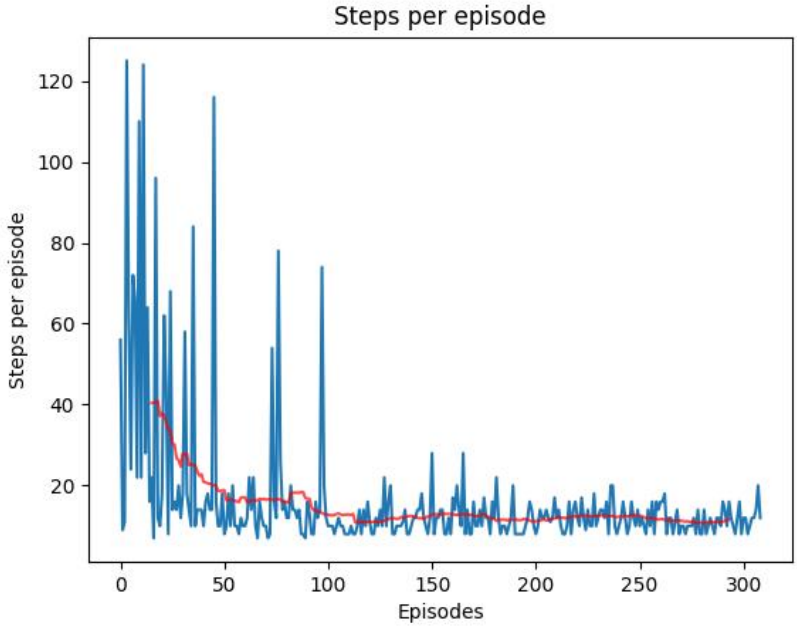


Figure 2. Passos por episódios

4.1. Variando a taxa de aprendizado

Para determinar como a taxa de aprendizado influencia a execução do algoritmo , foram feitas duas execuções com os mesmos parâmetro alterando somente a taxa de aprendizado

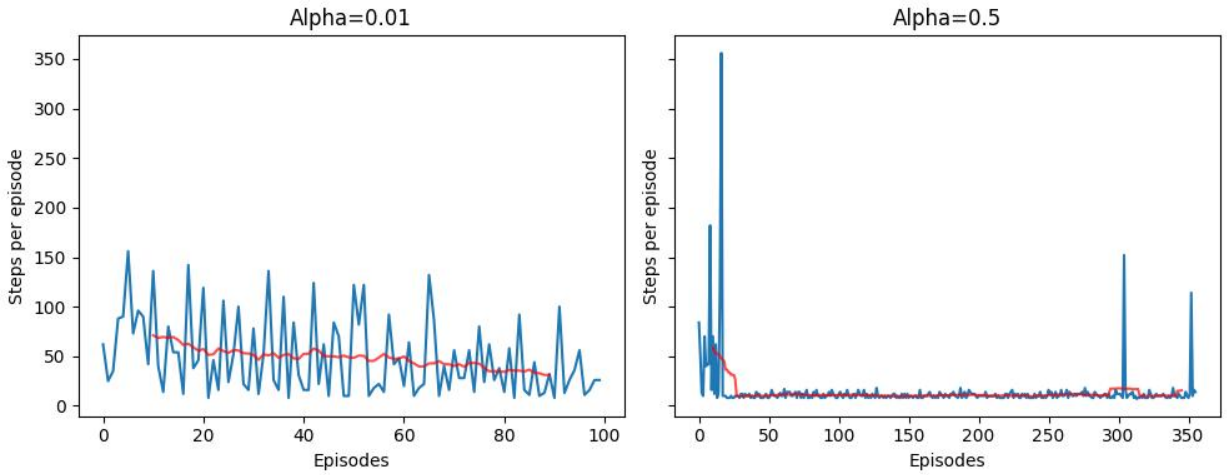


Figure 3. Passos por episódio variando a taxa de aprendizado

Observando os gráficos é possível perceber que a alpha=0.01 foi muito pequeno ,e o algoritmo não conseguiu convergir, enquanto 0.5 foi muito agressivo e causou picos no número de passos mesmo após 300 episódios de treinamento , portanto , o valor de 0.1 utilizado na figura 2 , empiricamente é mais apropriado .

4.2. Variando o epsilon-greedy

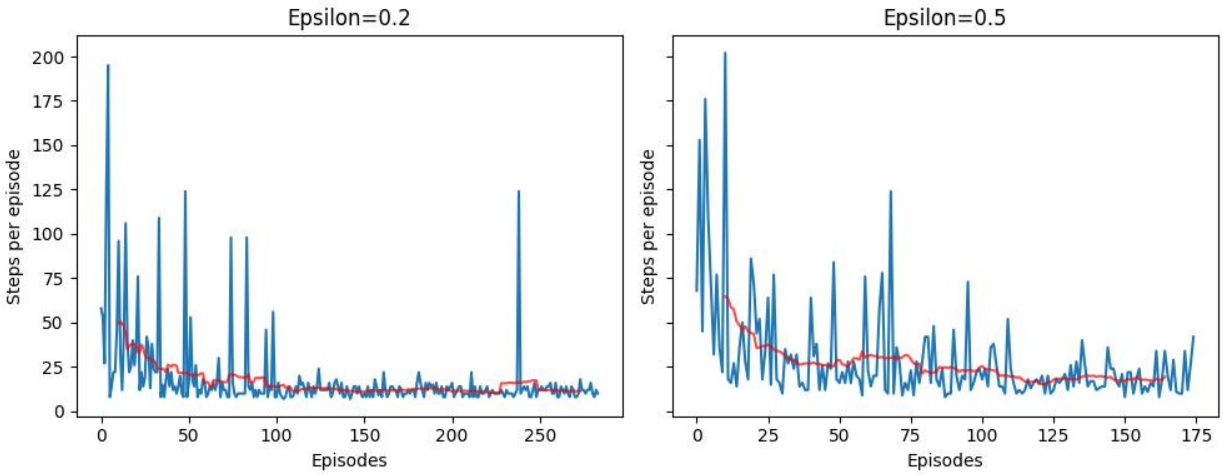


Figure 4. Passos por episódio variando ao e epsilon

Variando o epsilon é possível perceber que 0.5 causou muita aleatoriedade no algortimo e o desempenho foi prejudicado. Ainda, como existe uma slip rate dentro do Q-learning, já existe de certa forma uma aleatoriedade nas execuções mesmo quando episolon é igual a 0.

4.3. Variando a recompensa padrão

Quando o agente não está em um estado de vitoria ou derrota, ele ganha uma pequena recompensa negativa para desencorajar a exploração excessiva. Esse parâmetro é muito importante e influencia diretamente o tempo de convergência e a taxa de acertos do algoritmo.

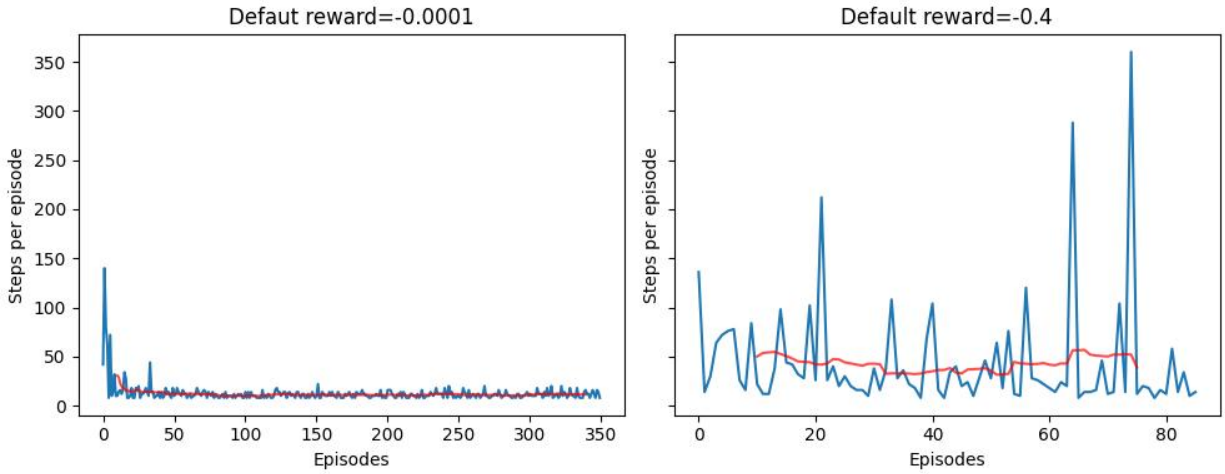


Figure 5. Passos por episódio variando a recompensa padrão

É possível perceber que -0.4 foi uma recompensa negativa ruim nesse cenário , uma vez que a unica recompensa positiva é a vitoria com valor +1. Dessa forma, é importante balancear as recompensas negativas e positivas dos estados para melhorar o aprendizado do Q-learning.

5. Evolução dos Q-values

Uma outra maneira de visualizar o aprendizado do Q-learning é observando os valores de Q-value. Dessa forma, foi feito o grafíco que para cada episódio , gera a média dos maiores Q-values da Q-table.

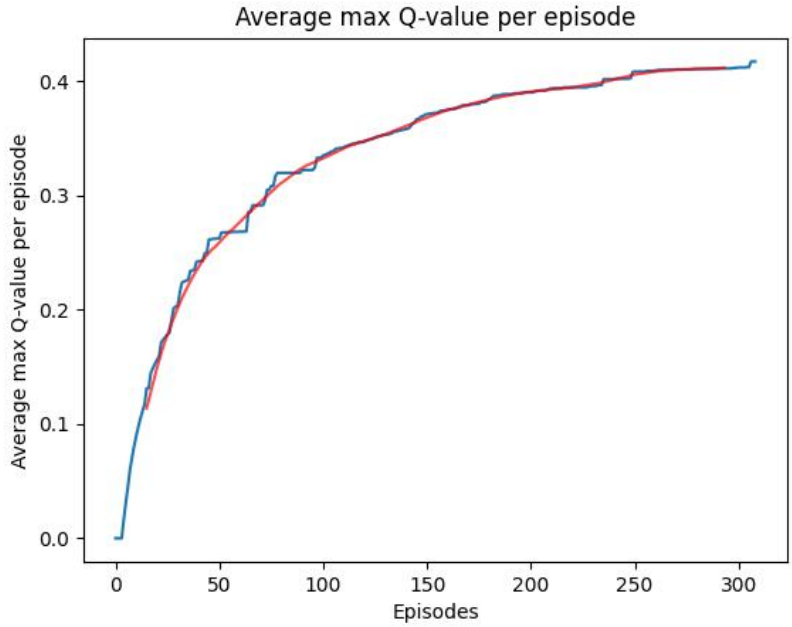


Figure 6. Média dos Q-values maximos

6. Evolução das recompensas totais

O Q-learning é um algoritmo que tenta maximar a recompensa total gerada por uma politica. Dessa forma, é importante visualizar como as recompensas de cada episódio variam de acordo com a execução do algoritmo.

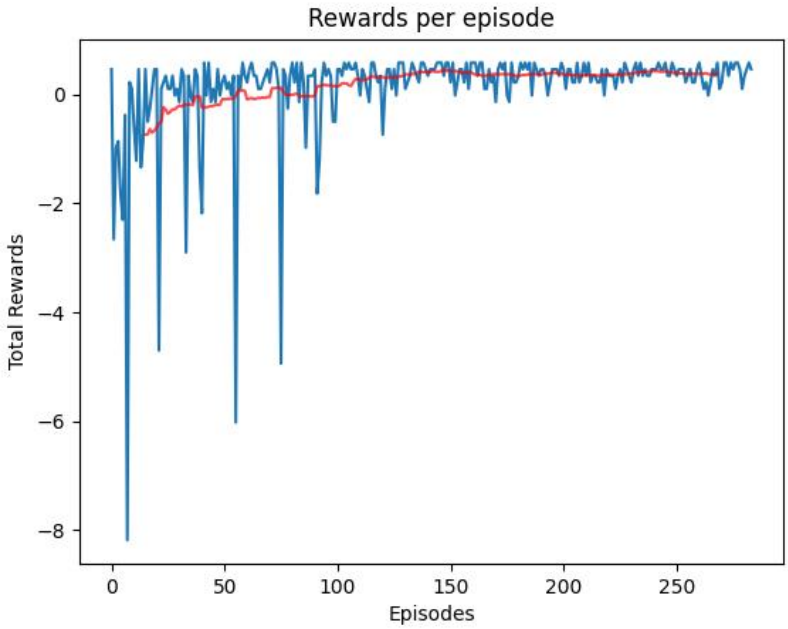
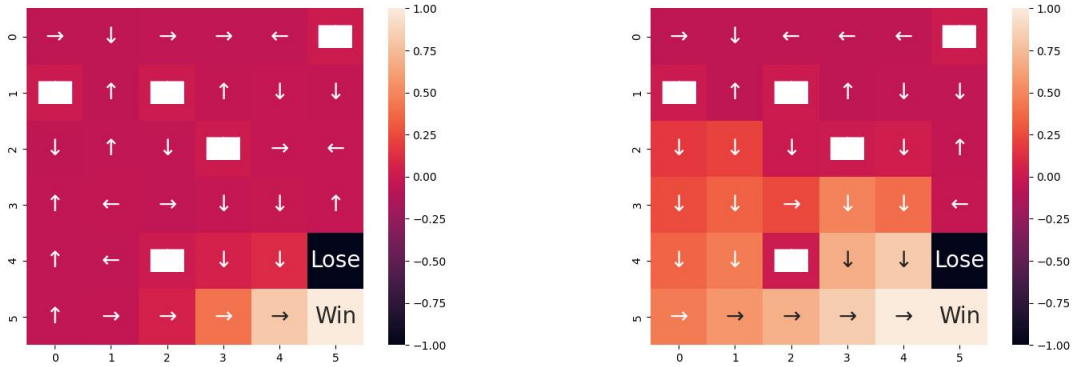


Figure 7. Recompensas totais por episódio

É possível perceber que existe um máximo para o valor da recompensa que representa quando o agente tomou uma política ótima.

7. Visualizando as Q-tables

Uma outra maneira de observar a evolução dos Q-learning durante o treinaimento é observar quais politicas as Q-tables geram.



(a) 500 passos

(b) 5000 passos

Figure 8. Evolução das Q-tables com o treinamento

É possível perceber que os valores da Q-table indicam politicas melhores a serem tomadas pelo agente em medida que o número de passos aumenta, assim, isso é um indicador que realmente ocorreu um aprendizado.

8. Utilizando o modelo treinado

Para conferir a qualidade da Q-table gerada foi feita uma função que simula o agente porém não altera a Q-table e confere quantas vezes o agente ganha e perde e toda vez que uma vitoria ou derrota acontece um novo estado inicial aleátorio é escolhido. Além disso, é contado o número de passos dentro de cada episódio, se uma certa execução passar de 300 passos, também é considerada uma derrota. O intuito é observar como o tempo de treinamento influencia a capacidade do modelo de achar o estado vitorioso a partir de qualquer estado possível.

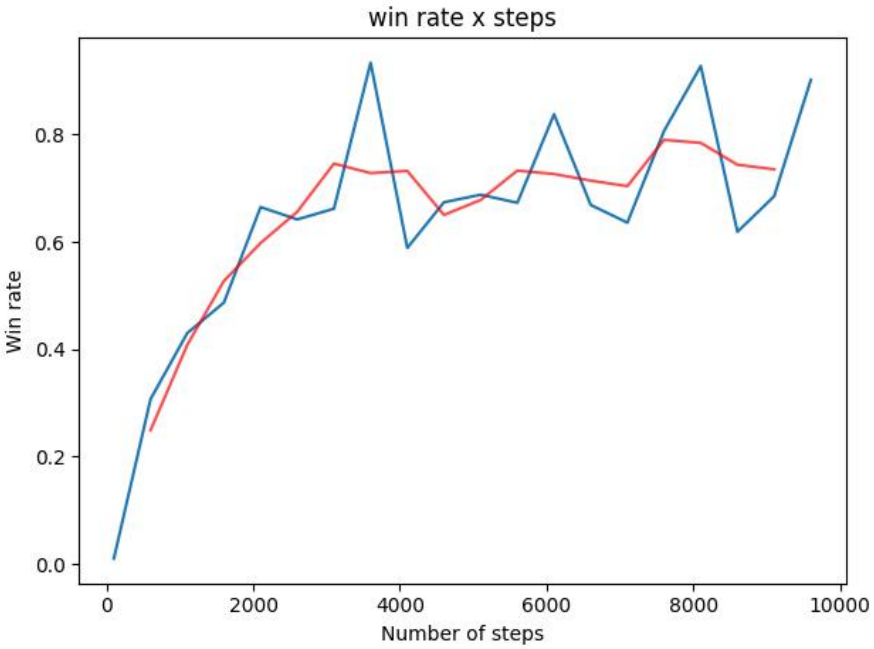


Figure 9. taxa de vitória x tempo de treinamento

Esse gráfico indica que com 2000 passos de treinamento o módelo conseguiu chegar no estado vitorioso com menos de 300 passos 45% das vezes e com 10000 pasos cerca de 75% das vezes.

9. Usando informações disponíveis para melhorar o aprendizado

Suponha que seja possível saber todas as posições iniciais possíveis no grid, dessa forma, poderíamos usar um reinicio aleatório do estado inicial ao fim de cada episódio.

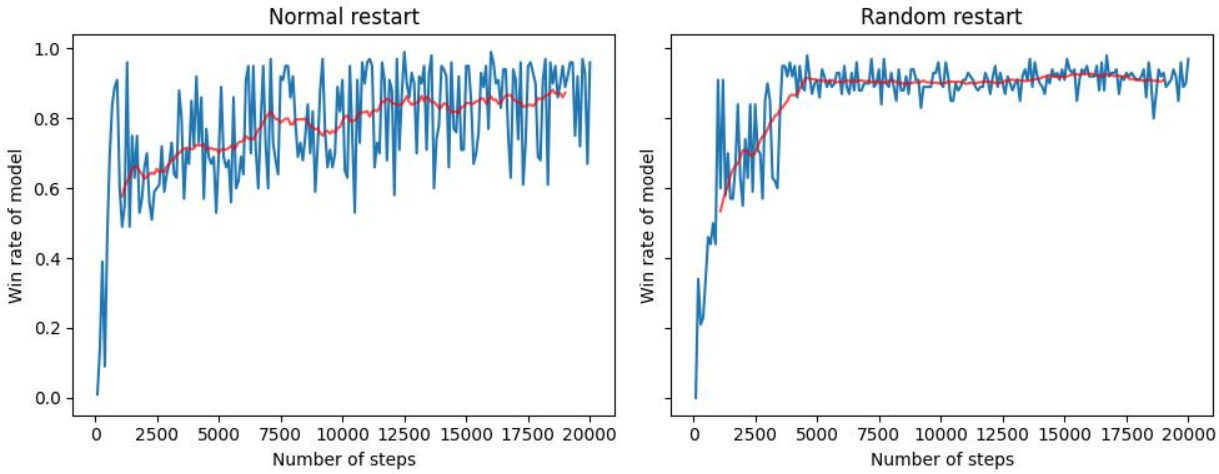


Figure 10. Comparação inicio normal x inicio aleatório

Observando a figura é possível perceber que o reinicio aleatório convergiu mais rápido e teve resultados mais satisfatórios do que o reinicio normal. È importante destacar que para cada simulação foram feitos 1000 episódios para remover a aleatoriedade da simulação em si, uma vez que ainda existe a slip rate.

10. Conclusão

Portanto, podemos concluir que os hiper parâmetros do Q-learning assim como o tempo de treinamento influenciam diretamente nos resultados obtidos como por exemplo na eficiência do algoritmo e no tempo de convergência da Q-table.Ainda, é possível utilizar informações disponíveis ou adquiridas durante o treinamento para melhorar os modelos e ter politicas mais próximas da ótima.

References

[1] Luiz Chaimowicz. *Slides da disciplina introdução à Inteligência Artificial, DCC642 – Aprendizado de máquina*. 2023.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach*. 4th. Prentice Hall, 2020.