Q-LEARNING POLICY FOR THE CLEANING ROBOT

Otimização de Estratégias Orientada por Dados

Tomás Branco Vicente
125604

Conteúdo

Introdução

Este projeto explora a aplicação do algoritmo de Q-learning para corrigir e otimizar a política de navegação do robô.

Os objetivos principais são:

1. Demonstrar que o Q-learning pode convergir de qualquer política inicial para uma ótima, através de testes empíricos.
2. Identificar os valores dos hiperparâmetros que aceleram a convergência do algoritmo.

Métologia

Ambiente de Simulação com a Classe GridWorld: Utilizámos a classe GridWorld para criar um ambiente de simulação que reflete o espaço operacional do robô de limpeza. Este ambiente é uma grelha 3x4 onde definimos estados, obstáculos e recompensas, permitindo simular diferentes cenários de navegação para o robôgridworld.

Scripts Python Utilizados:

1. *gridworld.py*- Define o ambiente de simulação onde o robô opera.
2. *qlearning.py*- Implementa o algoritmo Q-learning, que atualiza as políticas baseadas nas recompensas recebidas ao executar ações no ambiente GridWorld qlearning.
3. *main.py*- Orquestra o processo de treino e experimentação, utilizando os métodos definidos em qlearning.py e gridworld.py, gerindo múltiplas configurações de hiperparâmetros e registando os resultadosmain.
4. *analyze.py*- Analisa os resultados dos experimentos, produzindo visualizações e resumos estatísticos que ajudam a entender o desempenho das várias configurações de hiperparâmetros testadasanalyze.

Configuração dos Hiperparâmetros: Os hiperparâmetros foram essenciais para ajustar o processo de aprendizagem. Configurámos:

- $\alpha\ (Taxa\ de\ Aprendizagem)$: Influencia o quanto novas informações afetam o conhecimento existente.
- $\gamma\ (Fator\ de\ Desconto)$: Determina a importância das recompensas futuras na atualização do valor Q.
- $\varepsilon(Política\ Epsilon-Greedy)$: Equilibra entre exploração (escolher ações aleatórias) e explotação (escolher ações baseadas no conhecimento adquirido)

Configuração Inicial

Configuração Inicial do Ambiente A configuração inicial do GridWorld consiste em três matrizes principais que definem o comportamento e a dinâmica do ambiente de navegação do robô:

Matriz de Estado:

Representa o ambiente em termos de células transitáveis, obstáculos e estados terminais.

Valores possíveis:

- 0.0: Célula transitável.
- -1.0: Obstáculo, onde o robô não pode entrar.
- 1.0: Estado terminal (positivo ou negativo).

Configuração inicial:

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & -1.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Matriz de Recompensa

- Define a recompensa atribuída ao robô ao entrar em cada célula.
- Configuração inicial:

$$\begin{bmatrix} -0.04 & -0.04 & -0.04 & 1.0 \\ -0.04 & -0.04 & -0.04 & -1.0 \\ -0.04 & -0.04 & -0.04 & -0.4 \end{bmatrix}$$

Matriz de Política Inicial:

Define as ações iniciais recomendadas para cada estado, que serão ajustadas ao longo do treino.

Configurada aleatoriamente, com exceção de obstáculos e estados terminais, onde não há ações válidas.

Valores possíveis:

- 0.0: Mover para cima.
- 1.0: Mover para a direita.
- 2.0: Mover para baixo.
- 3.0: Mover para a esquerda.

- -1.0: Sem ação, estados terminais.
- NaN: Não aplicável, como em obstáculos.

$$\begin{bmatrix} 1.0 & 1.0 & 1.0 & -1.0 \\ 0.0 & \text{NaN} & 0.0 & -1.0 \\ 0.0 & 1.0 & 0.0 & 3.0 \end{bmatrix}$$

Hiperparâmetros

Configuração	α	γ	ε_{start}	Decay Step
Setup 1	0.01	0.9	0.2	50000
Setup 2	0.005	0.85	0.2	50000
Setup 3	0.1	0.95	0.3	100000
Setup 4	0.001	0.8	0.2	100000
Setup 5	0.005	0.8	0.1	25000

Tabela 1 - Hiper parâmetros

Setup 1 $(\alpha = 0.01, \gamma = 0.9, \varepsilon_{start} = 0.2, Decay\ Step = 50000)$

- Configuração base para aprendizagem estável com um fator de desconto alto e exploração moderada.

Setup 2 $(\alpha = 0.01, \gamma = 0.9, \varepsilon_{start} = 0.2, Decay\ Step = 50000)$

- Taxa de aprendizagem mais baixa para estabilidade, com um fator de desconto ligeiramente reduzido.
- Comparável ao Setup 1 para análise de impacto de α.

Setup 3 $(\alpha = 0.01, \gamma = 0.9, \varepsilon_{start} = 0.2, Decay\ Step = 50000)$

- Alta taxa de aprendizagem e fator de desconto para priorizar a aprendizagem rápido e recompensas a longo prazo.
- Exploração inicial mais alta para garantir diversidade de ações no início.

Setup 4 $(\alpha = 0.01, \gamma = 0.9, \varepsilon_{start} = 0.2, Decay\ Step = 50000)$

- Baixa taxa de aprendizagem e menor fator de desconto, adequados para cenários onde recompensas imediatas não são relevantes.

Setup 5 $(\alpha = 0.01, \gamma = 0.9, \varepsilon_{start} = 0.2, Decay\ Step = 50000)$

- Exploração inicial reduzida e *decay* rápido, ideal para ambientes mais previsíveis.

Treino com Q-learning

O treino foi realizado através da função train_q_learning, que seguiu as seguintes etapas:

1. Inicialização:

Em cada iteração (época), o robô é posicionado aleatoriamente em um estado válido.

A política e a matriz de valores QQQ são ajustadas com base nas interações com o ambiente.

2. Iterações por Época:

Durante cada época, o robô realiza ações até alcançar um estado terminal ou atingir o limite de 1000 passos.

A sequência de passos é guiada pela política atual utilizando uma estratégia epsilon-greedy:

Após cada ação:

- A matriz Q é atualizada com a fórmula de Q-learning
- A política é ajustada para refletir a ação ótima com base nos valores Q.
- O número de visitas a cada par estado-ação é registado.

Monitorização da Convergência:

- A estabilidade da política é avaliada calculando a diferença média entre a política atual e a política da época anterior.
- A convergência é assumida quando essa diferença permanece abaixo de um limite ($threshold = 1e^{-3}$) por pelo menos 10 épocas consecutivas.

Resultados

Setup	α	γ	ε_{start}	*Decay Step*	*Recompensa Média*	*Desvio Padrão*	*Convergência (épocas)*	*Estabilidade Final*	*Recompensa Final*
1	0.01	0.9	0.2	50000	0.7181	0.5417	275	0	0.88
2	0.005	0.85	0.2	50000	0.7177	0.5415	9	0	0.8
3	0.1	0.95	0.3	100000	0.7081	0.4296	9	0	0.84
4	0.001	0.8	0.2	100000	0.7077	0.5541	9	0	1
5	0.005	0.8	0.1	25000	0.7242	0.5347	23	0	0.8

Tabela 2 - Resultados por Hiper parâmetros

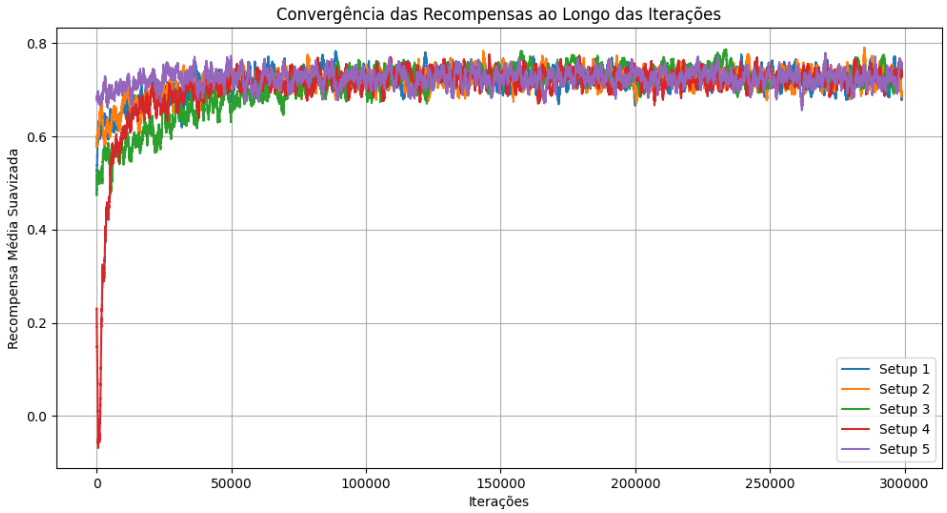


Figura 1 - Convergência das Recompensas ao Longo da iterações

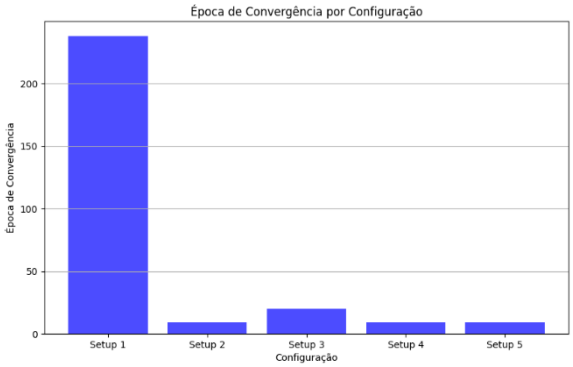


Figura 3 - Época de convergência Por Setup

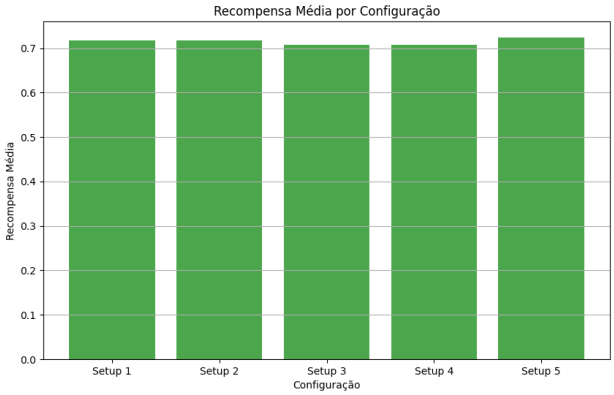


Figura 4 - Recompensa Media Por Setup

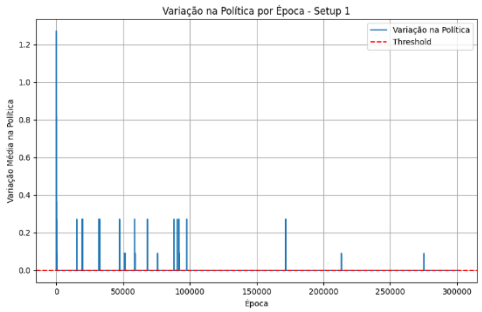


Figura 5 - Variação Setup 1

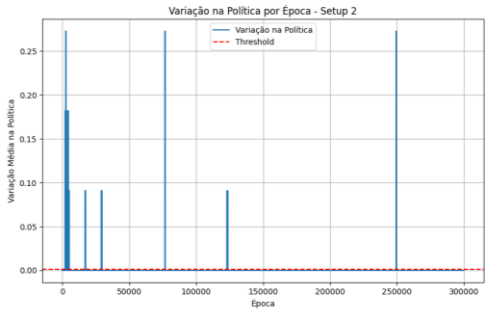


Figura 6 - Variação Setup 2

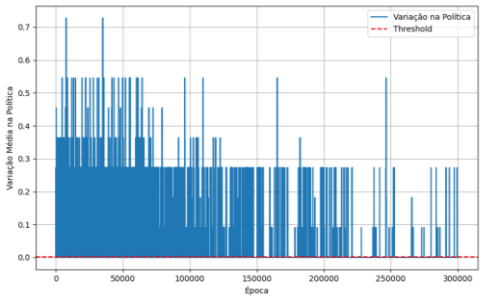


Figura 7 - Variação Setup 3

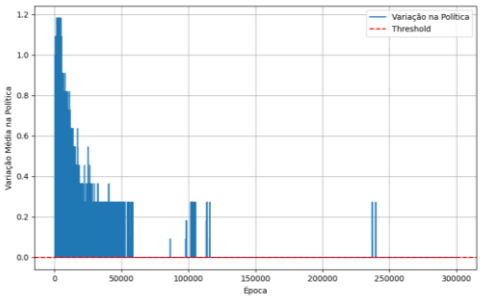


Figura 8 - Variação Setup 4

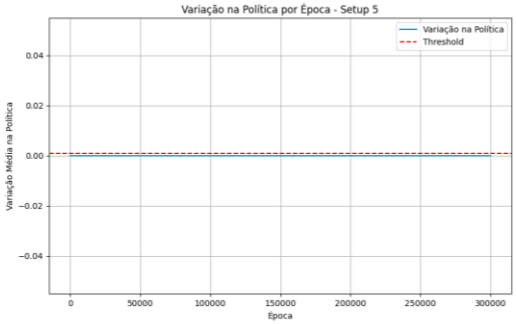
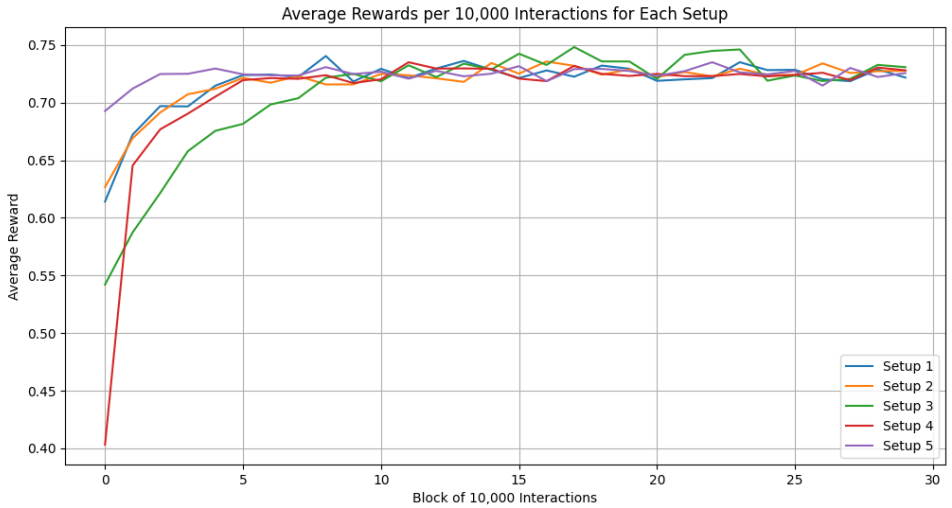


Figura 9 - Variação Setup 6



Observações

Recompensa Média

- Setup 5 apresentou a maior recompensa média (0.724193), sugerindo boa eficácia em alcançar estados desejados.
- Setup 4 teve a menor recompensa média (0.707718), potencialmente devido ao baixo γ, priorizando recompensas imediatas.

Época de Convergência

- Setups 2, 3 e 4 convergiram rapidamente (9 épocas).
- Setup 1 demorou significativamente mais (275 épocas).

Desvio Padrão

- Setup 4 apresentou maior variabilidade (0.554088), indicando maior instabilidade nas recompensas.
- Setup 3 apresentou a menor variabilidade (0.429553), sugerindo maior consistência.

Estabilidade Final

- Todos os setups alcançaram estabilidade perfeita, ou seja, convergiram na política.

Eficiência Inicial vs. Estabilidade

- Setups com taxas iniciais de crescimento mais rápidas (como o Setup 5) podem ser preferíveis em cenários onde a eficiência inicial é crítica.
- No entanto, o Setup 4 pode ser mais adequado para cenários que tolerem exploração inicial mais longa para priorizar decisões futuras.

Decisão Baseada no Custo por Iteração

- Se o custo por interação for alto, setups como o Setup 5 podem ser mais vantajosos devido à rápida estabilização da política.
- Em ambientes com baixo custo por interação, setups como o Setup 4 poderiam ser explorados para maximizar políticas de longo prazo.

Flexibilidade para Ambientes Dinâmicos

- O desempenho consistente de setups como o Setup 5 ao longo de múltiplos blocos pode indicar maior robustez em cenários dinâmicos.

Conclusão

Neste trabalho, demonstrámos empiricamente que o algoritmo Q-learning, aplicado ao problema do robô de limpeza, converge para a política ótima independentemente da política inicial adotada. Realizando múltiplas tentativas verificámos que todas as configurações testadas atingiram estabilidade, com a variação média entre políticas consecutivas reduzindo-se para valores abaixo do limite pré-estabelecido $(threshold = 1e^{-3})$.

Além disso, analisámos a influência dos hiperparâmetros na velocidade de convergência. Observámos que combinações equilibradas de α (taxa de aprendizagem), γ (fator de desconto) e ϵ (exploração inicial), juntamente com um passo de *decay* $(decay\_step)$ ajustado, foram cruciais para acelerar o treino. Em particular, os setups com:

- $Setup\ 2\ -\ \alpha = 0.005, \gamma = 0.85, \epsilon_{Start} = 0.2, decay\_step = 50000,$
- $Setup\ 5\ -\ \alpha = 0.1, \gamma = 0.95, \epsilon_{Start} = 0.3, decay\_step = 100000,$

mostraram-se mais eficazes, convergindo em apenas 9 épocas e apresentando recompensas médias elevadas (Setup 5: 0.724193, Setup 2: 0.717661).

Por outro lado, setups com valores extremos de α (muito baixos ou muito altos) ou γ (baixos) resultaram em menor eficiência e maior variabilidade nas recompensas. Por exemplo, o Setup 4 apresentou a menor recompensa média (0.707718) e maior desvio padrão (0.554088), indicando instabilidade e menor consistência.

Estes resultados destacam a importância de ajustar os hiperparâmetros de forma criteriosa, considerando as características específicas do ambiente e o objetivo de otimizar a aprendizagem.

Por fim, os resultados obtidos reforçam a robustez do Q-learning como um método fora da política para resolver problemas de decisão sequencial, mesmo em cenários com informações limitadas sobre o ambiente. A flexibilidade na configuração dos hiperparâmetros permite adaptar o algoritmo a diferentes necessidades, balanceando eficiência, estabilidade e exploração do espaço de estados.