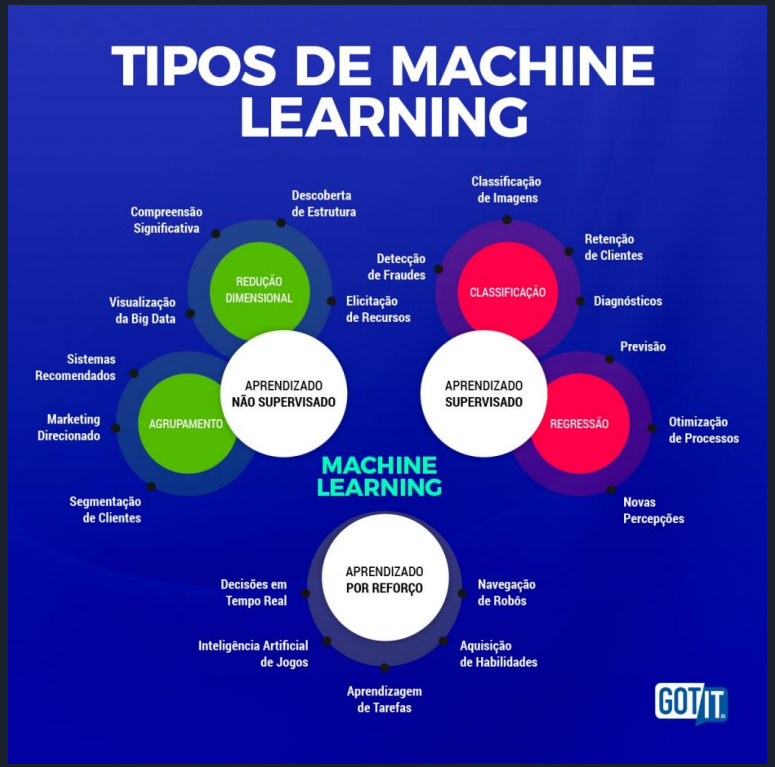
Treinando uma IA

minicurso

Profº: Marcos Campos

Introdução

Como vimos nos dias anteriores, existem algumas formas de Aprendizado de Máquina (*Machine Learning*).



Hoje vamos ver na prática como funciona o **Aprendizado por Reforço**, muito utilizado em jogos, simulações, robótica, navegação etc.

Nosso objetivo hoje é fazer com que uma IA ache o melhor caminho para sair de um labirinto.

Código do projeto no Github

Como funciona?

O Aprendizado por Reforço (*Reinforcement Learning - RL*), funciona através de um **Agente**, onde ele aprende a **tomar decisões** interagindo com um **Ambiente**. Em cada etapa, o **Agente** observa o estado atual do ambiente, escolhe uma **ação** e, como **resultado** recebe uma **recompensa (ou punição)** e passa a um novo estado. Com base nesse feedback, o agente **ajusta sua estratégia para ao longo do tempo** maximizar a recompensa acumulada.

Em resumo, é um processo de tentativa e erro: o agente experimenta diferentes ações, avalia os resultados e vai aprendendo quais ações levam a melhores resultados.

Algoritmos como **Q-Learning** e métodos baseados em gradiente de política são comumente usados para estimar o valor das ações ou diretamente a melhor política.

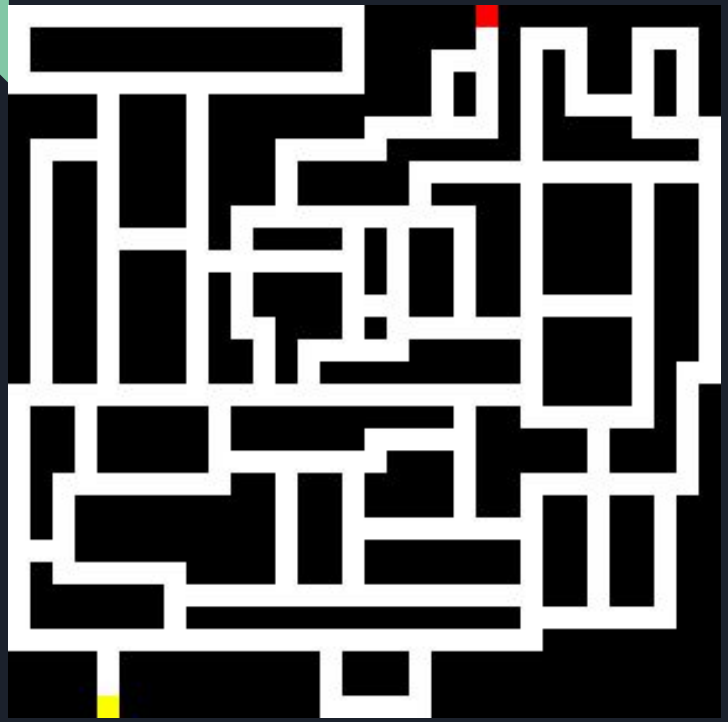
Q-Learning

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \cdot \max_{(a')} Q(s', a') - Q(s, a) \right]$$

Onde:

- α (taxa de aprendizado) controla o quanto novos valores influenciam os antigos;
- r é a recompensa imediata obtida ao executar a ação;
- γ (fator de desconto) pondera a importância das recompensas futuras;
- s′ é o novo estado após a ação;
- $\max_{(a')} Q(s', a')$ representa a melhor recompensa futura esperada a partir do novo estado.

Como irá funcionar nosso treinamento?



O Ambiente será um labirinto, onde o ponto amarelo representa a entrada, ponto vermelho a saída, e o branco representa os caminhos possíveis.

O labirinto tem formato 32x32 pixels, fique à vontade para editar da maneira que quiser.

Depois de aplicar as regras de movimentos possíveis *(cima, baixo, direita e esquerda)* e detecção de colisões. Vamos rodar a simulação aplicando o algoritmo Q-Learning. No final ele deve mostrar o melhor caminho encontrado, ou seja, o caminho que percorrerá menos pixels para chegar na saída.

Ferramentas e tecnologias utilizadas: Python, Pygame, Piskel

Inspirações

- Inteligência Artificial Aprendendo! - Universo Programado