Desenvolvimento de uma Inteligência Artificial que joga Campo Minado

Gabriel Kury Fonseca[1], **Luiz G. P. Mendes**[2], **Rafael de T. Navarro**[3]

[1]Faculdade de Computação e Informática – Universidade Presbiteriana Mackenzie (MACK)
São Paulo – SP – Brasil

{10390103,10382703,10389955}@mackenzista.com.br

Resumo. *Este artigo têm como objetivo a recriação do popular jogo "Campo Minado"em python, e então desenvolver uma inteligência artificial que aprende a vencer o jogo através da aprendizagem por reforço utilizando o Deep Q-Learning*

1. Introdução

Os jogos digitais são amplamente utilizados tanto para entretenimento quanto para desenvolvimento de habilidades cognitivas, como resolução de problemas e pensamento crítico. O campo minado, um clássico jogo de lógica, oferece um cenário ideal para testar algoritmos de inteligência artificial (IA). A complexidade do jogo, no qual o jogador precisa identificar as bombas baseando-se apenas em pistas numéricas, é um desafio interessante para IA, especialmente quando aplicada a técnicas de aprendizado por reforço.

Neste artigo, propomos o desenvolvimento de um sistema de campo minado em Python, onde uma IA aprenderá a jogar utilizando o Deep Q-Learning, para aprender por reforço. A partir de sucessivas partidas, a IA será capaz de otimizar suas estratégias para minimizar o risco de encontrar bombas e maximizar o número de casas seguras reveladas.

2. Descrição do Problema

O campo minado é jogado em um tabuleiro de dimensões pré-definidas, onde algumas células escondem bombas e outras contêm números que indicam quantas bombas estão nas casas ao redor. O objetivo do jogador é revelar todas as casas seguras sem clicar em nenhuma bomba. Porém, o jogador só possui informações parciais sobre o tabuleiro, o que adiciona uma camada de incerteza às suas decisões (KAYE, 2000).

O problema central abordado neste projeto é como uma IA pode aprender a jogar campo minado de maneira eficiente. A IA precisará desenvolver uma estratégia que minimize a incerteza e maximize a probabilidade de revelar células seguras. O aprendizado por reforço será utilizado para permitir que o algoritmo aprenda com suas ações, recompensando-o por movimentos corretos e penalizando-o quando uma bomba for ativada.

3. Dataset

Por ser um algoritmo de aprendizagem por reforço, nenhum dataset é usado neste trabalho. Porém, foi desenvolvido um campo minado em python para o trabalho. Tal desenvolvimento é abordado na subseção 4.2

4. Metodologia e Resultados Esperados

A metodologia proposta para este projeto pode ser dividida em três etapas principais: desenvolvimento do jogo, implementação da IA e avaliação dos resultados.

4.1. Estudo na literatura

O artigo escrito por KAYE (2000) descreve o jogo campo minado (a versão do windows), e o aborda do ponto de vista computacional. Porém tal abordagem não é feita no âmbito da IA, mas sim da complexidade. Nele o campo minado é classificado como np-completo.

O artigo escrito por CLIFTON; LABER (2020) aborda o Q-Learning desde seu início, sua aplicação em diferentes áreas, como na estatística e na IA, e fornece exemplos específicos de aplicação. O Q-Learning é uma fórmula apropriada para lidar com sequências que aprende uma política de tomada de decisão baseada na atualização do valor esperado de ações em estados específicos. Esses valores esperados consideram a recompensa imediata e a recompensa futura potencial, que dependem das decisões sequenciais do agente; ele é derivado da Fórmula de Bellman. Ao tratar do Q-Learning no campo da IA, é citado o deep Q-network (DQN), que é uma rede neural que estima o valor do Q-Learning. O uso da DQN possui diferentes vantagens, porém a mais importante delas é que o Q-Learning tradicional pode exceder os limites de memória em problemas complexos, tornando o uso de uma alternativa como o DQN essencial.

4.2. Desenvolvimento do Jogo

A primeira fase consiste na implementação de uma versão simples do campo minado em Python, utilizando a biblioteca random para gerar partidas diferentes. A configuração padrão será um tabuleiro de 80 células (8x10) com 10 bombas, que é baseada no Campo Minado do Google na dificuldade ”Fácil”. A lógica para gerar o campo, calcular as dicas numéricas e determinar as regras de vitória e derrota será desenvolvida baseada no já citado Campo Minado do Google. Em relação à disposição das bombas no tabuleiro, nenhuma regra foi encontrada, portanto ela é feita de forma 100% aleatória após a primeira jogada.

4.3. Implementação da IA

A IA de aprendizado por reforço é uma rede neural que estima o valor do Q-Learning. Inicialmente nenhuma rede neural seria utilizada, porém devido à complexidade do problema da resolução do campo minado, foi utilizada a deep Q-network(DQN).

O aprendizado por reforço envolve 4 elementos: A estrutura de recompensa que, fornece uma recompensa ou uma penalidade baseado na decisão tomada; O Agente, que é o tomador de decisões, no caso o computador/modelo; ambiente, que é o onde o Agente toma suas decisões, neste caso o campo minado; o estado, que consiste no estado atual do jogo, as ações a serem tomadas. A mudança de estado é chamada de transição. Quando o Agente joga o jogo, o estado vai se moldando, e ao se basear nos valor do estado, o Agente pode tomar decisões melhores.

O Q-Learning possui Q-Values em uma Q-Table, que é uma tabela na qual as linhas são os possíveis estados e as colunas as possíveis ações. O inicial de cada célula é tipicamente 0, e os valores são definidos por uma equação derivada da equação de Bellman; de forma resumida, a equação é o valor da recompensa mais o produto da maior

recompensa do próximo estado com um desconto. O desconto varia entre 0 e 1, sendo que o mais próximo de 0, mais seguras as decisões tomadas.

O hiper-parâmetro épsilon () varia de 0 a 1, e é a probabilidade de exploração. Quanto maior o épsilon, maiores as chances do modelo arriscar em ações nas quais ele não tem muita informação ao invés de tomar a ação com maior probabilidade de sucesso. No treinamento, tal parâmetro começa com 0.9, e decai ao longo das iterações.

Experience Replay é o armazenamento das experiências do Agente em um buffer e, durante o treinamento, as experiências são apresentadas aleatoriamente. Isso reduz a correlação entre as amostras, evitando overfitting e permitindo um aprendizado mais eficiente. Um buffer pequeno ainda pode levar ao overfitting, e um buffer grande pode gerar um underfitting por conta da diluição das experiências.

Double Deep Q-Learning Networks (DDQN) é o uso de dois modelos para reduzir o viés de superestimação de Q-Values. O modelo primário escolhe as ações, e o modelo-alvo avalia as ações escolhidas. Os parâmetros do modelo primário são periodicamente copiados ou combinados no modelo-alvo, garantindo estabilidade no treinamento.

4.4. Avaliação dos Resultados

Após a implementação da IA, o treinamento em 10 mil jogos é feito e em seguida o modelo joga 200 partidas como teste. A partir dos resultados do teste são feitas analises sobre a performance, o que ocasionou ela, e como melhorá-la.

5. Resultados

Em teoria, o treinamento do modelo e a replicação do campo minado foram feitos com sucesso, porém das 200 partidas teste, em nenhuma o modelo venceu.

6. Conclusão

Este trabalho tentou criar uma inteligência artificial que aprende a jogar campo minado através do aprendizado por reforço. Apesar da replicação do jogo e o treinamento de um modelo aparentemente certos, os os resultados esperados não foram alcançados, visto que o modelo jogou 200 vezes após o treinamento e perdeu todas. Os motivos que podem levar a este resultado são a estratégia de recompensa mal implementada e a rede neural mal feita. Acredita-se que a estratégia de recompensa tenha sido bem feita; já a rede neural foi feita de maneira simples para que a execução no Google Colab não levasse muito tempo, e portanto, este deve ter sido o problema.

7. Links externos

Repositório no Github: https://github.com/GKury/Projeto-IA

Vídeo no Youtube: https://www.youtube.com/watch?v=N9yitdlElsU

8. Referências

CLIFTON, J.; LABER, E. Q-learning: Theory and applications. Annual Review of Statistics and Its Application, Annual Reviews, v. 7, n. 1, p. 279–301, 2020.

KAYE, R. Minesweeper is np-complete. Mathematical Intelligencer, v. 22, n. 2, p. 9–15, 2000.