UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Simulando multi-agentes em jogos de tabuleiro por turnos, utilizando aprendizagem por reforço via Value-Decomposition Network e Gymnasium API

Jorge de Melo Feldmann

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

CAPA

Simulando multi-agentes em jogos de tabuleiro por turnos, utilizando aprendizagem por reforço via Value-Decomposition Network e Gymnasium API

Jorge de Melo Feldmann

USP - São Carlos

2023

Jorge de Melo Feldmann

Simulando multi-agentes em jogos de tabuleiro por turnos, utilizando aprendizagem por reforço via Value-Decomposition Network e Gymnasium API

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Fernando Santos Osório

USP - São Carlos

2023

DEDICATÓRIA

Dedico esse trabalho à minha mãe Vicentina Gomes de Melo Feldmann, ao meu pai Jorge Ubirajara da Silva Feldmann e ao meu irmão Ubirajara de Melo Feldmann, os quais sempre me apoiaram e foram suporte nos momentos mais difíceis, e pelos quais eu fiz desse trabalho uma honra às suas memórias.

AGRADECIMENTOS

Gostaria de agradecer especialmente o Professor Fenando Santos Osório, que me orientou neste enorme desafio, trazendo possibilidades para este trabalho e conhecimento para minha vida.

Agradeço também à Professora Solange Oliveira Rezende por ser uma luz nessa jornada, pelas palavras e esclarecimentos, mesmos nos meus momentos pessoais mais difíceis.

EPÍGRAFE

O jogo é fato mais antigo que a cultura, pois esta, mesmo em suas definições menos rigorosas, pressupõe sempre a sociedade humana; mas os animais não esperaram que os homens os iniciassem na atividade lúdica. Os animais brincam tal como os homens.

JOHAN HUIZINGA (1938)

RESUMO

FELDMANN, J. **Simulando multi-agentes em jogos de tabuleiro por turnos, utilizando aprendizagem por reforço via Value-Decomposition Network e Gymnasium API**. 2023. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

O mundo dos games é vasto, popular e em constante inovação, sendo um dos mercados onde a inteligência artificial aflora a passos largos. Sob esse contexto, neste trabalho foi desenvolvido um jogo baseado em turnos e tabuleiro que simula caçador e caça, onde foi implementado e analisado o aprendizado por reforço de comportamentos de um sistema multi-agente (com mais de um caçador e uma presa). Buscou-se um aprendizado que valorize a cooperação entre caçadores, na captura da presa. A implementação foi realizada usando o Simulador Gymnasium, com o agentes treinados com o algoritmo Value-Decomposition Network. Foi avaliado o desempenho do algoritmo, baseado em técnicas de Aprendizado de Máquina, treinado através do aprendizado por reforço e auto-treino, através do relato de um experimento, como se dá o processo de preparação, execução e avaliação da curva de aprendizado mediante o algoritmo proposto.

Palavras-chave: videojogo; jogo digital; aprendizado de máquina; inteligência artificial; tabuleiro; turnos; auto-jogo; aprendizado por reforço.

ABSTRACT

FELDMANN, J. **Simulating multi-agent turn-based board games using reinforcement learning via Value-Decomposition Network and Gymnasium API.** 2023. Final paper (MBA in Artificial Intelligence and Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

The game's world is vast, widespread, and in constant innovation, being one of the markets where artificial intelligence thrives. In this context, a turn-based board game was developed that simulates hunter and hunting, where the behavior reinforcement learning of a multi-agent system (with more than one hunter and one prey) was implemented and analyzed. Learning that values cooperation between hunters in capturing prey was sought. The implementation was implemented using the Gymnasium Simulator, with agents trained with the Value-Decomposition Network algorithm. The algorithm's performance was evaluated, based on Machine Learning techniques, trained through reinforcement learning and self-training, through the report of an experiment, how is the process of preparation, execution, and evaluation of the learning curve through the proposed algorithm.

Keywords: videogame; digital game; machine learning; artificial intelligence; board-game; turn-based; auto-play; reinforcement learning.

SUMÁRIO

1. Introdução

O ato de jogar no sentido lúdico, é perceptível em vários animais, incluindo seres humanos, como elabora JOHAN HUIZINGA, 1938, em sua obra Homo Ludens, como comportamento intrínseco e anterior à cultura. Na mesma obra somos apresentados ao conceito de "círculo mágico", uma forma de manifestação de jogo ou ritual, onde as regras da realidade são suspensas e substituídas por regras de uma realidade artificial (LINSER et al, 2008), em outras palavras, um videogame.

Conforme relata GRATEZ, 1981, a história dos videogames pode ser rastreada até o início da década de 1950, com o primeiro jogo feito num osciloscópio, o Spacewar. O motivo da criação de um jogo, como pesquisa acadêmica, pode ser a demonstração do poder computacional, mas outra motivação intrínseca dos primeiros games, senão a principal, era porque jogar é divertido e faz parte do ser humano.

Não é sem motivo que a indústria dos videogames movimenta bilhões de dólares todos os anos (GARTENBERG, 2020), mais até que o cinema e a música (QUAST et al, 2021) . É um dos fenômenos do século 21 que mais se destaca como mídia e forma de entretenimento. Jogos como World of Warcraft, de 2004 (WILLIAMS, 2019); GTA, de 2014 (BLEEKER, 2018); e Cyberpunk 2077, lançado em 2020 (BAILEY, 2021); este jogos possuem altíssimos investimentos, vendas exorbitantes, milhões de jogadores, entre outros números relevantes, sendo o videogame considerado referência como mídia de entretenimento e a nona forma de arte (FUNK, 2011).

Nesse contexto, toda uma indústria foi criada, com novas disciplinas surgindo a cada nova geração de plataformas de video-games (BETHKE, 2003). Entre estas disciplinas, destacam-se os principais tópicos deste trabalho: Inteligência Artificial (IA) e Aprendizado de Máquina (AM). A Inteligência Artificial faz parte dos videogames desde sua concepção (GRANT, E. F. LARDNER, R., 1952), enquanto que o Aprendizado de Máquina, ainda que tão antigo quanto o primeiro tópico, tenha florescido entre 2010 e 2021, em jogos clássicos e milenares como Go (SILVER et al, 2017), seculares como o xadrez (KNIGHT, 2020), até jogos mais novos como Dota (BERNER et al, 2019) e jogos de Arcade (BELLEMARE et al, 2012).

O objetivo de implementar o Aprendizado de Máquina em jogos, além de fator diferencial quanto à experiência de um produto, é também tentar generalizar a solução para questões do mundo real (BERNER et al, 2019).

Dada a relevância desta relação entre videogames, Inteligência Artificial e Aprendizado de Máquina, será detalhado um experimento sobre as disciplinas em voga, no que diz respeito à experiência proposta aos jogadores, que procuram nos jogos digitais experiências e desafios no nível de outros jogadores.

1.1. Motivação

Jogos que possuem capacidades de simular outros jogadores ou oferecem um desafio que se adapta aos mais experientes ainda é um dos obstáculos na indústria de jogos digitais. Desenvolver algoritmos de Inteligência Artificial, com capacidade de Aprendizado de Máquina (Machine Learning) é um desafio interessante e com muitas possibilidades de aplicações, requerendo, inclusive, a possibilidade de criar múltiplos oponentes com diferentes habilidades e níveis de jogabilidade.

Sistemas e algoritmos de IA orientados para jogos são, também, um campo cujas aplicações podem ser utilizadas em outros segmentos, como aceleração de cálculos em multiplicação de matrizes (FAWZI et al, 2022) ou mesmo para elaborar algoritmos de ordenação mais eficazes (MANKOWITZ et al, 2023).

Considerando os tipos de jogos baseados em turnos e tabuleiros de duas dimenões, é possível detectar padrões como ações classificadas e seus valores. Usualmente estes tipos de jogos possuem um estado corrente, e a seleção de uma ação “adequada” para uma determinada configuração, onde esta ação nem sempre pode ser “ensinada” através de algoritmos de Aprendizado de Máquina supervisionados e convencionais. Uma vez que podem existir múltiplas “boas respostas” para uma mesma situação, e também, devido ao fato de que só vamos poder avaliar a qualidade das escolhas após várias jogadas, com a evolução do jogo, nem sempre podemos fazer uso de métodos supervisionados tradicionais, onde podemos considerar outras abordagens (p.ex. aprendizado não-supervisionado, aprendizado por reforço e DRL Deep Reinforcement Learning [RUSSEL, R.; NORVIG, P., 20, 2009]).

Apesar de existirem algoritmos tradicionais da Inteligência Artificial que podem ser aplicados em jogos de tabuleiro por turnos, como por exemplo o algoritmo MiniMax (WINSTON, 1992), abordagens mais recentes baseadas em técnicas de Machine Learning, podem obter resultados bem interessantes, e assim obter “jogadores inteligentes baseados em Inteligência Artificial e Aprendizado de Máquina”, com diferentes níveis de habilidades e desempenho.

1.2. Questões de pesquisa e objetivos

Utilizando técnicas de Aprendizado de Máquina, a plataforma Gymnasium API e o algoritmo Value-Decomposition Network, pretende-se responder as seguintes questões de pesquisa: é possível aprender uma estratégia inteligente de cooperação multii-agente em um jogo de tabuleiro em turnos, usando o aprendizado por reforço ? É possível testar e validar esse tipo de experimentos com o algoritmo Value-Decomposition Network simulado na plataforma Gymnasium API ?

2. Base conceitual e estado da arte

2.1. Jogos por turnos em tabuleiros

Os jogos referenciados neste trabalho são os jogos digitais (videogames, GASI, 2016), e se baseiam em alguns aspectos do jogo de tabuleiro, do tipo damas e no xadrez (PRITCHARD, D.B., 1994).

O tabuleiro possui duas dimensões (2D), sendo um conjunto de colunas (marcadas por letras) e linhas (numeradas), onde o encontro de uma coluna com uma linha resulta em uma célula, nomeada pela respectiva letra e linha. Exemplo: a célula A1 é nomeada pela coluna A e pela linha 1. A quantidade de linhas e colunas é a mesma, atribuindo um aspecto quadrado ao tabuleiro, como uma matriz quadrada matemática.

Já o estilo de jogo consiste no enfrentamento de 2 equipes ou jogadores, por turnos, onde os jogadores revezam a tomada de decisões, sendo atribuído a cada um deles um personagem e suas ações.

O que diferencia os jogos aqui abordados do jogo de damas e xadrez são a variedade de movimentos possíveis por cada peça e os temas envolvidos. Tomemos como exemplo o jogo Stratego, que simula batalhas medievais (WOODS, STEWART, 2012):

1. A cada jogador é atribuído um personagem fictício. No exemplo, o personagem é um cavaleiro medieval, com armas pré-definidas (espadas, martelos, etc).
2. Todos os personagens possuem uma quantidade de pontos, denominada vida. A condição de vitória para um jogador é zerar a vida do personagem oponente ou, após um número pré-definido de rodadas máximas, ter pontos de vida superiores ao oponente.
3. Todos os personagens possuem os mesmos atributos associados, como pontos iniciais de vida, quantidade de movimento no tabuleiro, etc.
4. As ações podem ser dos mais variados tipos e são valoradas, no sentido de que cada ação possui características e valores quantitativos associados. Exemplo: a ação de movimento determina quantas células um personagem pode se deslocar continuamente, a partir do ponto onde a ação é tomada. As demais ações têm, pelo menos, um atributo principal, que determina sua cláusula de sucesso, e geralmente possui a quantidade de pontos de vida que são retirados do oponente (dano), em caso de sucesso. Um exemplo: no caso de combates medievais, podem ser ações do tipo "efetuar um ataque ligeiro com a espada", "ataque forte com o martelo", e cada uma

dessas ações possui variáveis associadas. Para o ataque ligeiro, o dano pode ser 2, Já o ataque forte pode ter um dano de 5. São permitidas uma ou mais ações, geralmente uma para movimento e uma para qualquer outra ação, em qualquer ordem, dentro de uma lista limitada de opções.

5. Cada ação é submetida aos atributos do oponente. Essa ação é dita como sucesso se os referidos pontos de vida do oponente são retirados.
6. Os jogos geralmente terminam após eliminação dos oponentes ou, após um número de rodadas, aquele que obtiver mais pontos de vida.

Ainda que com variações, são exemplos deste tipo de jogo: Combate (jogo de tabuleiro físico, versão brasileira de Stratego), Dungeons and Dragons (criado em 1974, jogo de RPG físico), Final Fantasy (criado em 1987, RPG de video game), Pokemon (videogame lançado em 1996, baseado em revista-em-quadrinhos japonesa), entre outros.

2.2. Inteligência Artificial (IA)

Para LUGER e STUBBLEFIELD, 2009, Inteligência Artificial é a área da computação voltada para a automação de comportamentos inteligentes. Winston (citado em FERNANDES, 2005, p. 2) define essa área como o estudo da computação que torna possível perceber, raciocinar e agir. Diz ainda, que IA é o estudo das ideias que permitem aos computadores serem inteligentes (FERNANDES, 2005, p.2).

A Inteligência Artificial é a capacidade que um sistema computacional tem de raciocinar, planejar, processar ideias, compreender linguagem e adquirir conhecimentos. O conceito de IA pode ser sintetizado na capacidade do homem em desenvolver sistemas computacionais, que são capazes de simular o raciocínio humano e de ser inteligente (FERNANDES, 2005; LORENZI; SILVEIRA, 2011).

As técnicas de IA processam os dados para criar relatórios gerenciais para melhor interpretação dos dados e informações. A IA muitas vezes está atrelada a sistemas maiores de Big Data e a grandes bancos de dados. Desta forma, diferentes técnicas de IA podem ser empregadas no desenvolvimento de sistemas de apoio à decisão.

As aplicações utilizam técnicas de IA, como classificação, regressão, detecção de desvios, regras de associação, padrões sequências, agrupamento e sumarização. Atualmente, grande parte destas aplicações requer a utilização de Algoritmos de Aprendizado de Máquina

para que possam ser criados modelos eficientes de tratamento de dados baseados no aprendizado.

2.3. Redes Neurais Artificiais (RNA)

As Redes Neurais Artificiais são modelos matemáticos inspirados nas Redes Neurais Cerebrais. As Redes Neurais Cerebrais são constituídas de milhares de neurônios interligados, direta ou indiretamente. Estes neurônios transmitem pulsos eletroquímicos entre si, difundem esses pulos através de excitação mútua e interpretam estes sinais para que nós possamos agir. A partir dos estudos realizados sobre a maneira de pensar e agir, foi constatado que a teoria de que o cérebro poderia ser comparado a um computador digital convencional estava equivocada. O processamento das informações no cérebro é realizado de forma distribuída, fortemente interconectada, e com alto grau de paralelismo entre as unidades de processamento (neurônios).

Levando em consideração tais conhecimentos sobre a capacidade das redes neurais, despertou-se um enorme interesse, em desenvolver as RNAs, que pudessem modelar as redes neurais biológicas, que segundo FAUSSET (1995), são sistemas de processamento de informações que tem como princípio alguma inspiração numa rede neural biológica. As RNAs têm sido desenvolvidas como generalizações de modelos matemáticos da cognição humana neural.

Os neurônios biológicos basicamente são compostos por Dendritos e Axônios. Dendritos são filamentos que recebem as informações de outros neurônios ou de entradas externas (sensores). Não tem conexão física com os outros dendritos, e a ligação entre os dendritos é realizada através dos pulsos eletroquímicos que são denominados de sinapses. A partir dos sinais recebidos pelos dendritos, o Corpo Celular (SOMA) realiza uma série de transformações e processos internos e, os resultados advindos destes processos são propagados aos outros neurônios da rede. O axônio propaga, através das sinapses, o resultado dos processos do corpo celular aos neurônios subsequentes.

O modelo de neurônio computacional se assimila ao neurônio biológico. Em sua estrutura são encontrados os seguintes elementos: o conjunto de sinapses, o somador e a função de ativação. Assim como no modelo biológico, a sinapse consiste na entrada dos sinais para o neurônio, porém neste caso existem pesos diferentes associadas às determinadas

entradas, em que serão realizadas as operações. É importante ressaltar que os pesos serão positivos caso a sinapse seja excitatória e negativa se ela for inibitória (BIANCHINI, 2001).

O somador realiza as operações de soma dos resultados advindos dos sinais, enquanto que a função de ativação ajusta e repassa os valores vindos do somador, limitando a amplitude de saída do neurônio.

De uma maneira geral, as RNAs são redes maciças de unidades de processamento, paralelamente distribuídas e constituídas. Estas redes armazenam os conhecimentos que são empiricamente adquiridos, e os disponibilizam para usos posteriores, se assemelhando ao cérebro humano. A rede é treinada através de processos que ocorrem em seu ambiente e os conhecimentos são armazenados através de pesos sinápticos (HAYKIN, 1999).

As RNAs se dividem em diversos modelos. Alguns exemplos mais conhecidos são: Perceptron, MLP e Rede Neural Recorrente. O Perceptron foi o primeiro modelo implementado de RNA. Generalizando, o Perceptron é composto por duas camadas, sendo a camada de entrada responsável por executar a função de repassar à camada seguinte as variáveis de interesse, sem executar nenhuma operação e a camada de saída possui um neurônio, que corresponde à saída desejada. O Perceptron atua através do aprendizado de função de tomada de decisão para classificar dois conjuntos linearmente separáveis (MEHROTRA KISHAN e MOHAN, 2000).

Perceptron Multicamadas (Multi Layer Perceptron - MLP) é uma rede que conecta Perceptrons em camadas, sendo geralmente adotadas redes de 3 camadas. De maneira superficial, uma MLP pode ser vista como uma extensão natural do Perceptron, na qual se coloca outras camadas, que não têm ligação com as camadas de entrada e saída. A principal diferença do MLP é que os neurônios que são inseridos na RNA têm como função de ativação uma função do tipo não linear, ou seja, é aplicável a problemas independentes das classes serem linearmente separáveis. Uma de suas principais aplicações está relacionada ao reconhecimento de padrões (BIANCHINI, 2001).

A Rede Neural Recorrente se diferencia das redes MLP por ter, pelo menos, um loop de realimentação, ou seja, esta rede possui um sistema de feedback. Pode haver uma única camada de neurônio, onde apenas um neurônio reabastece as entradas dos demais neurônios (ARBIB, 2002).

As RNAs têm uma ampla gama de áreas de aplicação, como por exemplo, militares, processamento de imagem e reconhecimento de voz, controle de processos, entre outras. Devido ao bom nível de eficiência das RNAs no reconhecimento de padrões, foi proposto o seu uso para colaborar na solução do problema deste trabalho.

2.4. Redes Neurais Profundas (RNP)

Para BENGIO, (2009), as decisões e o reconhecimento de padrões que antigamente eram tomadas de forma subjetiva, atualmente os computadores conseguem resolver com as técnicas de AM. Computadores possuem facilidades na resolução de problemas com regras bem definidas como cálculos matemáticos, jogos de tabuleiro ou cartas, porém têm grandes dificuldades em atividades como reconhecimento de padrões em imagens, músicas, textos, etc. Por conta disso, as dificuldades enfrentadas sugerem que sistemas de IA precisam da capacidade de adquirir seu próprio conhecimento, aprendendo a extrair padrões de dados brutos e aprender a reconhecer padrões complexos.

Como as técnicas convencionais de AM são limitadas no processamento de dados brutos para o reconhecimento de padrões, elas dependem muito das técnicas de extração de features que transformam os dados brutos em uma representação eficaz, com o intuito de facilitar o trabalho do classificador de inferir os padrões nos dados de entrada. Logo, o extrator de características (features) tem uma enorme influência no desempenho de algoritmos de AM. Outra dificuldade nessas aplicações é que muitos fatores influenciam na variação dos dados observados.

As Redes Neurais Profundas (ou Deep Learning) é uma subárea do AM, formada por múltiplos níveis e camadas de pré-processamento (daí a origem do adjetivo "profunda") e posterior reconhecimento de padrões, ao contrário das RNAs originais, que eram rasas e compostas por poucas camadas.

As RNPs permitiram que modelos computacionais, compostos de múltiplas camadas de processamento, aprendessem representações de dados com vários níveis de abstração. Esses métodos melhoraram drasticamente o estado-da-arte em várias áreas, como descoberta de novas drogas e pesquisa do genoma. Através das RNPs é possível descobrir estruturas complexas em grandes conjuntos de dados usando o algoritmo de backpropagation e as Redes Convolucionais Profundas (RCP ou CNN - Convolutional Neural Nets) para indicarem como uma máquina deve alterar seus parâmetros internos, que são usados para calcular a representação em cada camada da representação anterior.

Algumas estruturas de RNPs tais como Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Deep Reinforcement Learning (DRL) e Convolutional Neural Networks (CNN) vêm sendo aplicadas em diversas áreas, obtendo resultados muito melhores do que os resultados indicados até então, em competições de AM e reconhecimento de padrões.

2.5. Treinamento de RNAs: Aprendizado de Máquina (AM)

A principal característica da inteligência humana é o aprendizado, essa representa os meios básicos para a obtenção de conhecimentos. Segundo HUA, 2009, o processo de aprendizagem humana envolve a memória, o pensamento, a percepção, o sentimento, e outras atividades mentais relacionadas. Comparado à aprendizagem humana, o Aprendizado de Máquina (AM) é mais rápido, o acúmulo de conhecimento é facilitado e os resultados de aprendizagem são mais fáceis de demonstrar. Entretanto, esse processo depende diretamente da ação humana, isto implica que todo o progresso do ser humano no campo de AM, vai aumentar a capacidade dos computadores em aprender, auxiliando no melhoramento do processamento das informações.

Segundo HUA, 2009, o aprendizado é uma atividade que processa a informação de fora pra dentro. As informações do ambiente externo são processadas e geram conhecimento, que posteriormente é armazenado em um repositório, guardando muitos princípios gerais que norteiam uma parte da ação de execução. O aprendizado dependerá muito da qualidade dessas informações fornecidas pelo ambiente, tornando a aprendizagem fácil e organizada ou difícil e desordenada.

Segundo ABRAMSON, (1963), as máquinas vão identificar e atribuir os objetos e elementos às classes, definindo um reconhecimento automático de padrões. Esse estudo atua na compreensão dos conceitos de como ocorre AM, uma das principais técnicas utilizadas na última década.

Segundo MITCHEL, (1997), BISHOP, 2006, p. 3 e outros autores, existem alguns tipos básicos de AM, a saber:Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Reforço. O **Aprendizado Supervisionado (AS)**, para cada exemplo apresentado ao algoritmo de aprendizado é necessário apresentar a resposta desejada (ou seja, um rótulo informando a que classe o exemplo pertence, no caso de um problema de classificação de imagens, por exemplo, como distinguir imagens de gatos e de cachorros). Cada exemplo é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. O objetivo do algoritmo é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados. Para rótulos de classe discretos, esse problema é chamado de classificação e para valores contínuos como regressão. Esse método de aprendizado é o mais utilizado.

Já no **Aprendizado Não Supervisionado (ANS)**, segundo GOODFELLOW et al, 2016, p. 105 , os exemplos são fornecidos ao algoritmo sem rótulos. O algoritmo agrupa os exemplos pelas similaridades dos seus atributos. O algoritmo analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters. Após a determinação dos agrupamentos, em geral, é necessária uma análise para determinar o que cada agrupamento significa no contexto problema sendo analisado.

No **Aprendizado por Reforço (AR)**, (MITCHEL, 1997, p. 367) o algoritmo não recebe a resposta correta, mas recebe um sinal de reforço, de recompensa ou punição. O algoritmo faz uma hipótese baseado nos exemplos e determina se essa hipótese foi boa ou ruim. O aprendizado por reforço é, geralmente, modelado como um modelo de Markov, onde na teoria da probabilidade, é um modelo estocástico usado para modelar sistemas de mudança pseudo-aleatória. Presume-se que os estados futuros do sistema dependem apenas do estado atual, não dos eventos que ocorreram antes dele.

Vale mencionar a técnica de AR empregada por SILVER et al, 2017, do **Auto-Jogo (Self Play ou SP)**: esta técnica consiste em realizar um treinamento que não possua interação humana em todo o processo, onde as partidas foram realizadas pelo programa contra si mesmo, sem intervenção de agentes humanos.

2.5.1. Aprendizado por reforço de multi-agentes

Segundo SUNEHAG et al, (2017), em comparação com a tarefa de ambiente de agente único, as tarefas de aprendizado por reforço com multi-agentes (MARL, multi-agent reinforcement learning) geralmente enfrentam um ambiente mais inconsistente devido ao comparecimento de outros agentes. A decisão de cada agente será baseada nas ações de outros agentes. Portanto, se aplicarmos um algoritmo de aprendizado por reforço de agente único, o desempenho do treinamento será instável e não será fácil obter um ótimo desempenho.

No aprendizado por reforço com múltiplos agentes, definimos k agentes onde as ações e observações são distribuídas entre eles. No problema MARL padrão, o ambiente subjacente é modelado como um jogo de Markov onde as ações são escolhidas e executadas simultaneamente, e novas observações são percebidas simultaneamente como resultado de uma transição para um novo estado.

Ainda que os agentes tenham suas próprias observações e apliquem suas próprias ações, cada agente recebe a ação conjunta para se otimizar e buscar otimizar a recompensa conforme definido acima.

2.5.2. Value-Decomposition Network

Para lidar com esse desafio, a “fatorização” tem sido amplamente estudada no domínio do aprendizado por reforço multiagente. Esta classe de problemas de aprendizagem é complicada devido aos grandes espaços combinados de ação e observação. Nas abordagens totalmente centralizadas e descentralizadas, encontramos o problema de recompensas espúrias e um fenômeno que chamamos de problema do “agente preguiçoso”, que surge devido à observabilidade parcial.

Para evitar que os “agentes preguiçosos” recebam altas recompensas, mesmo que não façam nada, o algoritmo Value-Decomposition Network aprende a decompor a função de valor da equipe em funções de valor do agente.

Essa rede visa aprender uma decomposição de valor linear ideal a partir do sinal de recompensa da equipe, propagando de volta o gradiente Q total por meio de redes neurais profundas que representam as funções de valor dos componentes individuais. Essa decomposição de valor aditivo é especificamente motivada por evitar os sinais espúrios de recompensa que surgem em aprendizes puramente independentes. A função de valor implícito aprendida por cada agente depende apenas de observações locais e, portanto, é mais facilmente aprendida (SUNEHAG et al, 2017).

2.6. Estado da arte

Em termos de Inteligência Artificial aplica a jogos, um trabalho de destaque foi desenvolvido por SILVER et al (2017), cuja aplicação no milenar jogo de Go levou ao desenvolvimento da rede AlphaGo, da startup DeepMind (agora pertencente ao Google), uma RNP considerada o melhor jogador do mundo. AlphaGo acompanhou e analisou mais de cem mil partidas de amadores de alto nível, vencendo o melhor campeão europeu e posteriormente, num desafio de melhor de 5, ganhou 4 partidas do então campeão mundial de Go. Em 2021, a AlphaGo já consegue generalizar e jogar também, de forma desafiadora, o xadrez e o shogi (um típico xadrez japonês).

Temos ainda a RNP OpenAI Five (BERNER et al, 2019), da famosa startup OpenAI, concorrente da DeepMind. A OpenAI Five venceu o melhor time do mundo no jogo DOTA 2, jogo de relevância mundial, onde dois times, cada um composto por cinco jogadores, realizam embates em uma arena. O jogo é famoso por permitir que times do mundo inteiro possam participar e cujas finais, The International, recebem milhões de espectadores todos os anos, desde 2011. A The International 2021 contou com aproximadamente 2,75 milhões de espectadores e com prêmios de mais de 40 milhões de dólares. A OpenAI conta com inúmeras soluções de AI, com exemplos de jogos clássicos como Pong, Space Invaders, Lunar Lander, entre outros.

O ALE: Arcade Learning Environment (BELLEMARE et al, 2012), rede treinada por AR, que foi treinada nas primeiras versões com jogos escolhidos arbitrariamente e depois testada contra jogos não treinados antes, performando de forma muito satisfatória na clássica plataforma de videogames Atari 2600.

Um trabalho mais recente envolvendo AR e Value-Decompostion Network foi realizado por KIM et al (2023), onde múltiplos algoritmos de MARL foram testados utilizando-se o jogo StarCraft, jogo de estratégia reconhecido mundialmente.

Este trabalho toma como referência a biblioteca OpenAI Gym (BROCKMAN et al, 2016), que foi criada para padronizar e acelerar o desenvolvimento de AR em jogos. A OpenAI Gym agora é conhecida como Gymnasium e mantida pela entidade Farama desde outubro de 2022 (FARAMA, 2022).

3. Desenvolvimento

Aplicações de Aprendizagem por Reforço (AR) visam especificamente situações onde é possível remover o ser humano como agente participante do treino. Neste sentido, a biblioteca Gymnasium fornece uma estrutura para treinar modelos de AR para atuar como agente jogador em treinos de jogos.

Para este trabalho, será analisado um jogo em turnos em tabuleiro que foi adaptado utilizando-se um código em linguagem de programação Python, e preparado um experimento de IA, usando-se treinos e testes pela técnica de AR e da plataforma Gymnasium API.

3.1. Metodologia

O conceito central do Aprendizado por Reforço é que um agente automático interaja com um ambiente por meio de observações e ações. Ao interagir com o ambiente, o agente pode receber recompensas (positivas ou negativas) e as usa para aprender e influenciar a tomada de decisão futura. Este tipo de abordagem apresenta elementos comuns.

O **agente** é o tomador de decisão, é aquele que opera a aplicação, que aplica algum processamento em dada entrada e devolve uma saída, dentro de um sistema. Para este trabalho, os agentes são cada jogador, dentro de cada time.

O sistema onde o agente é inserido é o **ambiente**, espaço metafórico que apresenta dados, na forma de observações, que servem como entrada e repercute a saída, na forma de ações do agente. Neste trabalho, o ambiente é um jogo, dado por um tabuleiro, onde agentes são posicionados. Existem turnos, em que cada agente pode escolher ações. Existe uma quantidade de ações a serem tomadas, cada uma com consequências previamente definidas. O jogo se desenrola entre os turnos e o resultado entre cada turno é avaliado. Tal como um jogo que se encerra quando objetivos são atingidos, o ambiente é encerrado quando os estados de vitória são avaliados como verdadeiros.

Observação são as variáveis que o agente faz sobre o ambiente, que pode ser usado para decidir quais ações tomar. Embora haja uma riqueza de dados diferentes que possamos apresentar ao agente para observar, as variáveis que foram consideradas para esta versão do jogo foram os elementos mais comuns e próximos aos critérios de vitória em jogos do mesmo estilo como, por exemplo, o posicionamento e integridade de cada jogador.

Ações são processamentos que um agente pode realizar sobre os estados, possibilitando alterá-los. Quando o agente escolhe agir, com base na exploração do melhor

padrão aprendido até agora, ele decide, por exemplo, qual inimigo atacar no jogo ou qual a melhor direção a ser tomada, ou mesmo não realizar nenhum ação. Já o **espaço de ações** são as ações possíveis.

Um **passo** ou step é a menor unidade de momento de um ambiente, em que as variáveis do ambiente são passadas ao agente e o mesmo devolve as alterações a serem aplicadas no ambiente. No interim desta relação, algoritmos são aplicados para determinar as ações possíveis e as consequências das escolhas das mesmas.

A **recompensa** é a utilidade que o agente recebe por realizar as ações “desejadas”. Assim, os estados informam ao agente em que situação ele está atualmente, e as recompensas sinalizam os estados aos quais ele deve aspirar. O objetivo, então, é aprender uma “política”, algo que diga qual ação tomar em cada estado para tentar maximizar a recompensa.

Um **episódio** é uma sequência de estados, ações e recompensas, que termina com o estado terminal. Neste trabalho, um jogo inteiro pode ser considerado como um episódio, sendo o estado terminal alcançado quando uma das seguintes condições são verdadeiras: um time foi eliminado (vitória para o time que permanece e derrota para o eliminado) ou a quantidade limite de rounds é atingido.

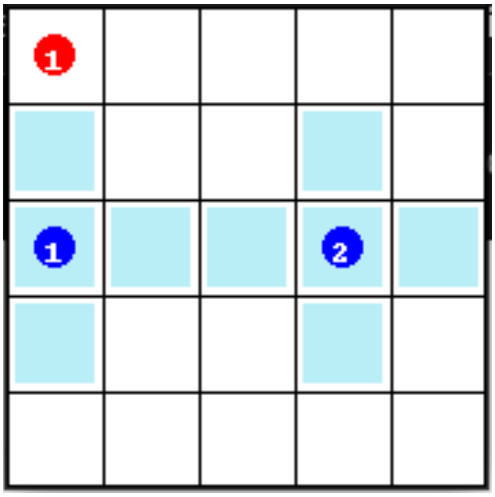
3.2. O jogo

O código de jogo escolhido é o Caça e Caçador (ANURAG, 2019), em sua versão PredatorPrey5x5-v1. O jogo apresentou uma baixa complexidade em seu código-fonte, com menos de 400 linhas de código em um único arquivo Python, o que facilita o entendimento de detalhes de implementação. Além disso, já estava integrado com a OpenAI Gym, e possui comentários e referências técnicas (como wiki) que ajudam no processo de experimentação.

O **tabuleiro** simples é simulado, envolvendo duas equipes opostas em um tabuleiro quadrado, cujas dimensões sejam iguais tanto em colunas ou linhas.

A primeira das equipes é denominada "caça", é constituída por jogador programado através de heurística simples. A outra equipe é denominada "caçadores", cujos jogadores serão os agentes deste trabalho, conforme a Figura 1.

Figura 1 - Imagem da inicialização do jogo, apresentado a caça (em vermelho) e os caçadores (em azul), com suas áreas de captura.



A cada turno, os jogadores podem executar uma dos ações no seguinte espaço de ações: mover para uma célula em uma das quatro direções cardeais ou não fazer nada.

A caça é programada com uma heurística simples, em que suas escolhas consistem em se locomover para longe dos caçadores ou, caso não os veja, de forma aleatória, incluindo não se movimentar. A caça enxerga apenas uma célula adjacente, nos pontos cardeais.

Uma recompensa positiva é dada apenas se vários caçadores capturarem uma caça simultaneamente, exigindo cooperação. Os caçadores recebem uma recompensa de "prey_capture_reward" se pelo menos dois capturarem uma caça ao mesmo tempo, mas recebem recompensa negativa "penalty" se apenas um caçador capturar a caça.

O alcance de visão e captura de um caçador é de uma célula adjacente, nos quatro pontos cardeais.

Cada jogador tem acesso às suas observações, sendo suas próprias coordenadas e seu próprio identificador. Os caçadores têm acesso às observações uns dos outros e também observam as coordenadas de caças em seu raio de visão.

O **objetivo dos episódios** é o sucesso dos caçadores ao capturar a caça. A “captura” de uma caça é definida quando a mesma está dentro do alcance de captura de, pelo menos, um caçador. Uma caça é considerada morta se for capturada por mais de um caçador e é retirada do jogo. A condição final desta tarefa é quando todas as caças são mortas (vitória do time dos caçadores) ou um máximo arbitrário de passos no ambiente (vitória da caça), denominado "max_steps".

3.3. Preparação do experimento

O algoritmo de aprendizado utilizado neste trabalho foi o Value-Decomposition Networks (SUNEHAG et al, 2017), usando-se a implementação de ANURAG (2021), mesmo autor do código do jogo. Sua implementação em Python também é de razoável compreensão, contendo pouco mais de 200 linhas em um único arquivo.

Executando-se o arquivo do algoritmo de aprendizado, o mesmo é inicializado com o ambiente do jogo. O algoritmo foi testado localmente e ajustes foram feitos para que o mesmo utilizasse a nova versão da Gymnasium API.

Para obter a curva de aprendizado, rastrear os episódios e poder avaliar o processo como um todo, foi utilizada a plataforma online WEIGHTS AND BIASES (2023), que oferece tais recursos de forma gratuita.

Para depurar erros, visualizar como os episódios acontecem e a evolução do modelo, foi reaproveitada a interface gráfica da Gymnasium API. Para fins de registro e análise, uma saída gerando vídeo dos episódios foi acoplada.

Em testes iniciais, foi verificado que um episódio dura menos de 0,1 segundo. Para os propósitos desse projeto, assumiu-se um valor arbitrário total de 4 (quatro) horas para o experimento como um todo.

Dentro dessas horas (240 minutos), foi estipulado que cada etapa ficaria com a metade (120 minutos) do total. Para cada etapa, uma janela de tempo de 10 minutos foi planejada da seguinte forma: o programa foi paralisado, e um vídeo foi gerado desta janela de tempo, para que anotações fossem feitas, e o programa continuaria de onde estava parado, até atingir o limite de tempo de cada etapa, mantendo desta forma as variáveis e enriquecendo cada etapa. A diferença é que na etapa de treino os parâmetros eram atualizados e na etapa de teste nenhum parâmetro foi alterado.

Logo, temos um resultante de um total de 240 minutos, previstas janelas a cada 10 minutos para análise e anotações, num total previsto de 24 análises e anotações, divididos igualmente na metade, sendo então 12 para cada etapa.

Os atributos que foram levados em consideração para a análise foram: a curva de aprendizado, a quantidade de recompensas e a quantidade de vitórias de cada equipe.

Os seguintes atributos foram configurados de forma arbitrária:

- Tamanho do tabuleiro: 5 x 5.
- Quantidade de caças: 1.
- Quantidade de caçadores: 2.
- Máximo de passos "max_steps" foi configurado para 100.
- Penalidade por capturar sozinho "penalty": -0,5.
- Penalidade por cada passo sem captura "step_cost": -0,01.
- Recompensa por capturar a caça de forma conjunta "prey_capture_reward" : 5.

Lembrando que os agentes são a equipe dos caçadores, as obsevações que os agentes tem acesso são uma lista, constituída das coordenadas do agente, seu identificador, as coordenadas da caça, concatenadas com as observações do outro agente.

As duas equipes foram inicializadas da seguinte maneira: os caçadores foram posicionados de forma aleatória no tabuleiro. Já a caça é inserida em uma das células que não está no alcance de captura dos caçadores.

Para os testes, mantendo-se as mesmas configurações do treino, porém removendo-se o aprendizado: o modelo não mais “aprenderia”, mantendo seus parâmetros, ainda que as recompensas fossem emitidas para análise.

3.4. Observações a priori

Após configuração e testes iniciais, algumas percepções apareceram.

Primeiramente, quando os agentes não se movem, para o total máximo de passos, a cada passo sem captura, um deles recebe uma penalidade de -0,01. Para um total de 100 passos máximos, temos -1 de recompensa para cada agente, totalizando -2 para cada episódio. Ou seja, cada episódio com recompensa total de -2 significa que em nenhum passo a caça passou pelos caçadores. Essa observação foi nomeada de **fracasso máximo dos agentes por inércia**, recompensa do episódio de = -2.

Logo, foi imaginado um caso ideal de captura, onde a caça está no canto e cada caçador em seus flancos, ainda sim um passo é necessário para que a) ambos os caçadores andem em direção à caça e pelo menos um deles a tenha no campo de visão e b) a caça não terá movimento disponível no próximo passo, ainda sim será desconto de -0,01, resultando em captura por ambos de 5 para cada, 10 no total, porém descontado -0,01 do passo, numa pontuação máxima possível de 9,99. Essa obsevação foi nomeada **sucesso máximo dos**

agentes, recompensa do episódio determiante de 9,99. Ou seja, sempre será deduzido -0,01 de todos os episódios e a maior recompensa possível é 9,99.

Entre esses extremos, foi hipotetisado que o menor sucesso possível seria a captura no lance imediatamente antecedente ao fracaso máximo dos agentes por inércia ou seja, atingindo-se o total de passos, onde a recompensa de -2, deduzida de -0,01, que é o lance proposto, resultando em -1,99. Caso a captura fosse realizada por ambos neste lance, cada agente recebe 5 pontos, num total de 10 pontos de recompensa, teríamos um total resultante no episódio de 8,01. Esta observação foi nomeada **sucesso mínimo sem captura prévia pelos agentes**, recompensa do episódio possível (mas não determinante) de 8,01.

Porém, quando apenas um caçador captura a caça, temos uma penalidade de -0,05. Caso o episódio todo seja repleto de capturas por apenas um caçador, sem que ambos capturem a caça ao mesmo tempo, num máximo de 100 passos, teríamos um total de -5 por agente, resultando num total de -10 Essa observação foi nomeada **fracasso máximo total de cooperação dos agentes**, recompensa do episódio possível (mas não determinante) de -10.

Com estas observações, pode-se deduzir que a menor recompensa possível, caso ambos os agentes capturem a caça seja a menor recompensa possível de -10 + 9,9, resultando em -0,01, denominada **sucesso mínimo com captura prévia pelos agentes**, recompensa possível (mas não determinante) de = -0,01.

Logo, pode-se concluir que episódios com recompensas:

- Menores que -0,01 certamente são derrotas para os caçadores.
- A partir de -0,01 é possível (mas não determinante) de que houve uma vitória por parte dos caçadores.
- A partir de 8,01 são certamente vitórias por parte dos caçadores.

3.5. Configuração da máquina local de testes

A configuração da máquina local onde os experimentos foram realizados é a seguinte:

- Modelo de notebook: Lenovo IdeaPad L340-15IRH Gaming
- Memória:16 GB
- Processador: Intel® Core™ i7-9750H CPU @ 2.60GHZ x 12
- Placa gráfica: NVIDIA Corporation GP107M [GeForce GTX 1050 3 GB Max-Q]
- Capacidade de disco: 1 TB SSD
- Sistema operacional: Ubuntu 22.04.2 LTS
- Arquitura: 64-bit

4. Análise e discussão dos dados

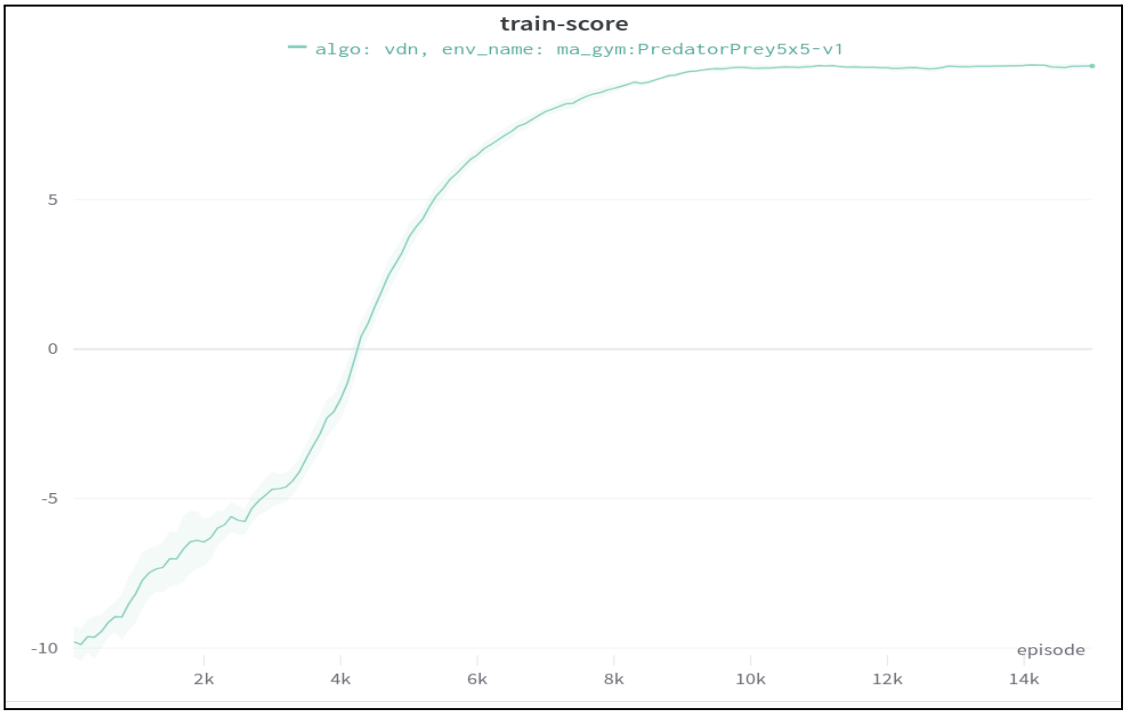
Durante o treino, nos primeiros 2 minutos, os agentes perderam todas as partidas sem capturar a caça nenhuma vez.

A partir da quinta janela de análise (5 minutos após o início), notou-se que o modelo começou a reagir, começando capturar pelo menos uma vez a caça. A partir daí, a evolução do modelo foi progressiva, e a primeira vitória do modelo ocorreu no episódio de número 4299. A partir daí, o modelo começou a ser vitorioso, a cada 2 partidas em média.

A partir do episódio de número 7813, após cerca de 20 minutos, o modelo tornou-se imbátivel, vencendo todas as partidas. As observações tomadas mostravam que o modelo organizava uma espécie de padrão, onde cada caçador iniciava sua ronda, cada um para um lado do tabuleiro e, ao detectar a caça, ambos de dirigiam a ela diretamente.

Decidiu-se paralisar arbiratrariamente em 15000 episódios para que as anotações pudessem ser finalizadas e trazidas para este trabalho. O total da execução do treino ficou em torno de 43 minutos e o registro das análises levou cerca de 30 minutos.. Como o tempo total foi menor que o previsto, o novo critério de parada da etapa de teste seria, 15000 episódios ou 2 horas, o que fosse atingido primeiro.

Figura 2 - Curva de aprendizado e recompensas durante o treino



Repetiu-se o mesmo procedimento para as 2 horas seguintes. Estranhamente, o modelo começou o teste perdendo por absoluto **fracasso máximo dos agentes por inércia** nos primeiros 100 episódios.

As primeiras possibilidades de vitória vieram a partir do episódio 200. A partir do episódio 300 os agentes saíram do negativo, ou seja começaram a capturar a caça de forma solitária e, desde então, o modelo evoluiu numa ascendente de recompensas, até o episódio 1499.

A curva de aprendizado mostrou variações entre vitórias e derrotas entre os episódios 1500 e 3500, e partir daí o modelo não perdeu mais contra a equipe programada por heurísticas. A cada janela de análise ficou mais claro a predominância do modelo e a hipótese de que nada mudaria então. Nenhum outro padrão foi detectado, dado que o modelo manteve as mesmas estratégias do treino.

Entre o episódio 14199 e o 14299 foi atingida a maior recompensa de 9,892.

O episódio de número 15000 foi atingido numa execução total de 32 minutos. Cada episódio se manteve na duração de menos de 0,1 segundo.

Figura 3 - Curva de aprendizado e recompensas durante o teste

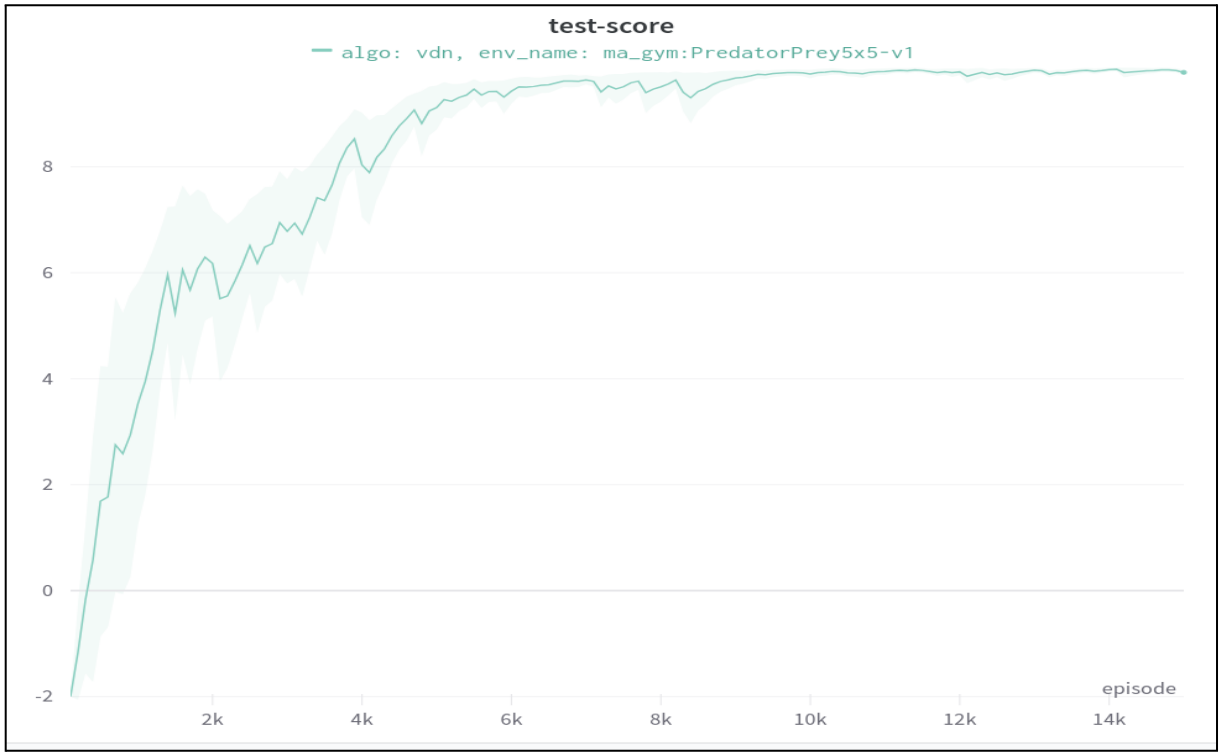


Tabela 1 - Média das recompensas por etapa

Etapa	Tempo total de execução	Total de episódios	Média das recompensas
Treino	43m 3s	15000	9,683
Teste	32m 15s	15000	9,47
Total	1h 15m 18	30000	N/A

Este experimento foi relatado e disponibilizado no endereço de internet, no Github: https://github.com/jotafeldmann/tcc-ia-big-data.

5. Conclusão

Respondendo-se à pergunta inicial, foi possível recriar um experimento, simulando um jogo multi-agente de tabuleiro por turnos, onde os agentes foram capazes de trabalhar de forma cooperativa, vencendo a maior parte das partidas em sua fase de testes, mediante uma equipe adversária programada por heurísticas, e realizar uma análise de desempenho do modelo de aprendizado e entedimento de como isso pode ser feito. Destacam-se as percepções e observações adquiridas durante a análise e as estratégias que foram detectadas visualmente nas análises durante o experimento, uma vez que não havia nada previsto a priori.

Um aspecto importante que está além do escopo deste trabalho, e foi percebido durante o processo, é a possibilidade de comparar outros modelos de aprendizado dado o ambiente padronizado que a Gymnasium API proporciona.

Por fim, foi revelador e lúdico poder colocar em prática o conhecimento aprendido, ao elaborar um experimento e verificar como é possível utilizar aprendizado de máquina para simular um jogo.

REFERÊNCIAS

RUSSEL, R.; NORVIG, P. **Artificial Intelligence: A modern Approach.** Englewood Cliffs, Prentice Hall, 1132p. ISBN-13: 9780137505135, 2009.

WINSTON, P. H. **Artificial Intelligence**. (3rd. edition) Addisons-Wesley Publishing, 737p. ISBN-13: 978-0201533774, 1992.

BETHKE, E. **Game development and production**. Texas: Wordware Publishing, Inc. ISBN 1-55622-951-8. 2003.

HUIZINGA, J. **Homo Ludens**. Routledge & Kegan Paul Ltd., 1938.

GRANT, E. F. LARDNER, R. **The Talk of the Town – It**. The New Yorker. 2 de agosto de 1952. Disponível em <https://www.newyorker.com/magazine/1952/08/02/it>, acessado em 30 de janeiro, 2022.

BELLEMARE, M.G. et al. **The arcade learning environment: An evaluation platform for general agents**. Journal of Artificial Intelligence Research. Cornell University, arXiv:1207.4708, June 18. 2013. Disponível em: https://jair.org/index.php/jair/article/view/10819/25823.

FAUSSET, L. **Fundamentals of neural networks architectures, algorithms, and applications.** Pearson. ISBN-13: 978-0133341867, 1993.

BIANCHINI, A.R. **Arquitetura de redes neurais para o reconhecimento facial baseado no neocognitron.** UFSCAR, p. 142, 2001. Disponível em: https://repositorio.ufscar.br/handle/ufscar/391

HAYKIN, S. **Redes neurais: Princípio e prática**. (2ª edição) Bookman. ISBN-13: 978-8573077186, 2003.

MEHROTRA KISHAN; MOHAN, C. K. R. S. **Elements of artificial neural networks Massachusetts**. p. 344, 2000.

ARBIB, M.A. **The handbook of brain theory and neural networks**. p, 1290, 2002.

LUGER, G. F.; STUBBLEFIELD, W.A. **Artificial Intelligence: Structures and Strategies for Complex Problem Solving.** 6th Edition, 2009.

FERNANDES, A.M.R. **Inteligência Artificial: noções gerais**. Florianópolis: Visual Books, 2005.

LORENZI, F.; SILVEIRA, S. R. **Desenvolvimento de Sistemas de Informação Inteligentes**. Porto Alegre: UniRitter, 2001.

HUA, W.; CUIQIN, M.; LIJUAN, Z. **A Brief Review of Machine Learning and its Application**. Information Engineering and Computer Science. ICIECS. International Conference on, vol., no., p. 1-4, 2009.

ABRAMSON, N. BRAVERMAN, D.; SEBESTYEN, G. **Pattern recognition and machine learning**. Information Theory, IEEE Transactions on, vol.9, ed. 4, p. 257-261, 1963.

MITCHELL, T. **Machine Learning**. S. l.: McGraw Hill, 1997.

BISHOP, C.M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Springer-Verlag, Berlin, Heidelberg, 2006.

GOODFELLOW, I; BAGGIO, Y. COURVILLE, **A. Deep Learning (Adaptive Computation and Machine Learning series)**. The MIT Press; Illustrated edition, 2016.

SILVER, D. et al. **Mastering the game of Go without human knowledge**. Nature, vol. 550, 19 de outubro, 2017. Disponível em: <https://www.nature.com/articles/nature24270.epdf?author_access_token=VJXbVjaSHxFoctQQ4p2k4tRgN0jAjWel9jnR3ZoTv0PVW4gB86EEpGqTRDtpIz-2rmo8-KG06gqVobU5NSCFeHILHcVFUeMsbvwS-lxjqQGg98faovwjxeTUgZAUMnRQ>.

BERNER, C. et al. **Dota 2 with Large Scale Deep Reinforcement Learning**. Cornell University, arXiv:1912.06680, 13 de dezembro, 2019. Disponível em: <https://arxiv.org/abs/1912.06680>.

BENGIO, Y. **Deep Learning architectures for AI.** Foundations and trends, 2009.

GARTENBERG, C. **US video game spending hit a 10-year high in June**. The Verge. July, 2020. Disponível em: https://www.theverge.com/2020/7/17/21328298/us-video-game-spending-high-npd-group-amount. Acesso em: 28 Jan. 2022.

QUAST, J., BRUNING, C., DEO, S. **This Opportunity for Investors Is Bigger Than Movies and Music Combined.** NASDAQ. October, 2021. Disponível em: https://www.nasdaq.com/articles/this-opportunity-for-investors-is-bigger-than-movies-and-music-combined-2021-10-03. Acesso em: 30 Jan. 2022.

BLEEKER, E. **GTA 5 Sales Hit $1 Billion, Will Outsell Entire Global Music Industry.** Fool. Created September 28, 2013, updated October 5, 2018. Disponível em: https://www.fool.com/investing/general/2013/09/28/gta-5-sales-hit-1-billion.aspx. Acesso em: 30 Jan. 2022.

BAILEY, D. **Cyberpunk 2077 sold 13.7 million copies in 2020, over half on PC**. Oct 13, 2021. Disponível em: https://www.pcgamesn.com/cyberpunk-2077/pc-sales. Acesso em: 30 Jan. 2022.

KNIGHT, W.. **Defeated Chess Champ Garry Kasparov Has Made Peace With Deep Blue AI.** Feb 21, 2020. Disponível em: https://www.wired.com/story/defeated-chess-champ-garry-kasparov-made-peace-ai. Acesso em: 30 Jan, 2022.

FUNK, J. **Games Now Legally Considered an Art Form (in the USA)**. May 6, 2011. Disponível em: http://www.escapistmagazine.com/Games-Now-Legally-Considered-an-Art-Form-in-the-USA. Acesso em: 30 Jan. 2022.

WILLIAMS, M. **How World of Warcraft Was Made: The Definitive Inside Story of Nearly 20 Years of Development.** US Gamer. July 22, 2019. Disponível em: https://www.usgamer.net/articles/how-world-of-warcraft-was-made-the-inside-story. Acesso em: 30 Jan. 2022.

LINSER, L. et al. **The Magic Circle – Game Design Principles and Online Role-play Simulations**. July 4, 2008. Ed-Media. Disponível em: http://www.simplay.net/papers/MagicCircle-Linser-Lindstad-Vold08.pdf. Acesso em: 30 Jan. 2022.

GRAETZ, J. M. **The origin of Spacewar**. August 1981. Creative Computing. Vol. 6, no. 8. pp. 56–67. ISSN 0097-8140. Disponível em: https://www.wheels.org/spacewar/creative/SpacewarOrigin.html. Acesso em: 30 Jan. 2022.

PRITCHARD, D.B. (1994). **The Encyclopedia of Chess Variants.** Games & Puzzles Publications. p. 84. ISBN 978-0-9524142-0-9.

WOODS, STEWART (16 August 2012). **Eurogames: The Design, Culture and Play of Modern European Board Games**. p. 17. ISBN 9780786490653.

GASI, FLÁVIA. **Mapas do imaginário compartilhado na experiência do jogar: o videogame como agenciador de devaneios poéticos**. 2016. 310 f. Tese (Doutorado em Comunicação e Semiótica) - Programa de Estudos Pós-Graduados em Comunicação e Semiótica, Pontifícia Universidade Católica de São Paulo, São Paulo, 2016. Disponível em: https://tede2.pucsp.br/handle/handle/19565

FAWZI, A., BALOG, M., HUANG, A. et al. **Discovering faster matrix multiplication algorithms with reinforcement learning**. Nature 610, 47–53 (2022). Disponível em: https://doi.org/10.1038/s41586-022-05172-4

MANKOWITZ, D.J., MICHI, A., ZHERNOV, **A. et al. Faster sorting algorithms discovered using deep reinforcement learning**. Nature 618, 257–263 (2023). Disponível em: https://doi.org/10.1038/s41586-023-06004-9

BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER J., SCHULMAN J., TANG, J., ZAREMBA, W., **OpenAI Gym**, (2016), disponível em https://arxiv.org/pdf/1606.01540.

FARAMA FOUNDATION, **Announcing The Farama Foundation: The future of open source reinforcement learning**, publicação eletrônica disponível em https://farama.org/Announcing-The-Farama-Foundation em 25 out. 2022.

ANURAG, KOUL, **Ma-gym: Collection of multi-agent environments based on OpenAI gym,** (2019), disponível em https://github.com/koulanurag/ma-gym.

ANURAG, KOUL, **Minimal-marl: Minimal implementation of multi-agent reinforcement learning algorithms** (2021). Disponível em: https://github.com/koulanurag/minimal-marl.

WEIGHTS AND BIASES, **Experiment Tracking**, site digital disponível em https://wandb.ai/site/experiment-tracking, acessada em 30 de julho de 2023.

SUNEHAG, P. et al. **Value-Decomposition Networks For Cooperative Multi-Agent Learning**. arXiv:1706.05296 [cs], 16 jun. 2017, disponível em https://arxiv.org/abs/1706.05296.

KIM, M. et al. **The StarCraft Multi-Agent Exploration Challenges: Learning Multi-Stage Tasks and Environmental Factors Without Precise Reward Functions**. IEEE Access, v. 11, p. 37854–37868, 2023. Disponível em https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10099458.