UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas - Departamento de Ciência da Computação
Monografia em Sistemas de Informação - DCC - UFMG
Tipo de Pesquisa: Tecnológica

Aplicação de Algoritmos Genéticos para o Treinamento de Redes Neurais em Jogos Digitais

Lucas Soares Maia
Orientador: João Guilherme Maia de Menezes

Janeiro/2025

Abstract

Este relatório apresenta os resultados da aplicação de Algoritmos Genéticos (AGs) para o treinamento de Redes Neurais Artificiais (RNAs) em jogos digitais, uma técnica conhecida como Neuroevolução. O trabalho foca na implementação do algoritmo NeuroEvolution of Augmenting Topologies (NEAT), que evolui tanto a topologia quanto os pesos da rede neural. Experimentos foram conduzidos em quatro jogos com diferentes níveis de complexidade: Flappy Bird, Pong, Space Shooter e Snake. Os resultados demonstram a eficácia do NEAT tanto em jogos mais simples quanto em mais desafiadores, mesmo com a complexidade do espaço de busca e da representação do estado do jogo. O relatório detalha a metodologia, os resultados e análises com o objetivo de promover o estudo do potencial da neuroevolução em outros domínios.

Chapter 1

Introdução

Jogos são dinâmicas que vão além do entretenimento, também funcionam como ferramentas valiosas para criar analogias com problemas reais, simular situações complexas, estimular o pensamento crítico e fomentar a inovação. Além disso, as versões digitais permitem experimentar situações perigosas ou desafiadoras sem riscos reais, testar diferentes abordagens e aprender com os resultados. Os jogos digitais oferecem uma liberdade criativa e experimental sem precedentes, permitindo explorar o "e se" de maneira segura.

A Inteligência Artificial (IA) se tornou uma ferramenta essencial no desenvolvimento de jogos modernos, permeando diversos aspectos da experiência do jogador e permitindo a criação de mundos virtuais mais ricos e imersivos. Desde oponentes mais astutos e adaptáveis, com comportamentos complexos gerados por IA, até a construção procedural de cenários, a IA impulsiona a inovação nos jogos. Ela aprimora animações, otimiza o pathfinding, personaliza a dificuldade, elevando a interação e o realismo a novos patamares. Com a IA, os jogos transcendem a simples execução de código pré-programado e se transformam em ecossistemas complexos e responsivos, proporcionando experiências cada vez mais personalizadas e memoráveis.

Este trabalho explora a aplicação de algoritmos genéticos (AGs) no treinamento de redes neurais artificiais (RNAs) em jogos, visando criar agentes com desempenho aprimorado. Inspirados na evolução darwiniana, os AGs otimizam os parâmetros das RNAs, simulando a dinâmica dos replicadores, um conceito da biologia matemática e da teoria dos jogos evolutiva que descreve como a frequência de entidades replicantes (genes, ideias, estratégias, etc.) muda ao longo do tempo em uma população. Em essência, replicadores mais bem-sucedidos em um determinado ambiente, ou seja, aqueles com maior "fitness" (aptidão), tendem a se propagar mais rapidamente, aumentando sua proporção na população, enquanto replicadores menos bem-sucedidos diminuem em frequência. Assim, a dinâmica dos replicadores oferece um modelo para entender a evolução e a disseminação de traços em diversas áreas, desde a genética até a cultura e a economia. Essa abordagem, chamada neuroevolução, apresenta vantagens em relação a estratégias de treino como backpropagation, gradiente descendente e aprendizado por reforço, incluindo a capacidade de navegar em espaços de busca complexos, adaptar-se a ambientes dinâmicos e descobrir soluções inovadoras.

Com isso, esse trabalho investiga o potencial da neuroevolução para o desenvolvimento de agentes inteligentes em jogos, explorando como a otimização baseada em AGs promove comportamentos adaptativos e eficientes, de forma que essa estratégia possa ser usada em outros contextos para treinamento de Redes Neurais.

Chapter 2

Referencial Teórico

Este trabalho foi desenvolvido integrando os conceitos de Redes Neurais Artificiais (RNAs) e Algoritmos Genéticos (AGs), técnica essa conhecida como Neuroevolução. Para aplicar esses conhecimentos, foi utilizado o treinamento de agentes inteligentes para jogarem jogos digitais, assim observando-se o desempenho e principalmente como os agente iam evoluindo e explorando novas soluções a cada geração.

Além disso, neste trabalho também foi utilizado o NEAT - NeuroEvolution of Augmenting Topologies, no qual além de usar o AG para otimizar os pesos da rede Neural, ele também otimiza a própria topologia da rede, adicionando/removendo nós e conexões. As próximas subseções abaixo tratam de explicar melhor os conceitos e técnicas utilizados neste trabalho.

2.1 Redes Neurais Artificiais (RNAs)

RNAs são modelos computacionais inspirados no cérebro humano, compostos por camadas interconectadas de neurônios artificiais. Cada neurônio recebe sinais de entrada, processa-os através de uma função de ativação e produz um sinal de saída. As conexões entre os neurônios possuem pesos associados, que determinam a força da influência de um neurônio sobre outro. Durante o treinamento, é realizado um ajuste nesses pesos de modo que a rede neural possa mapear padrões e realizar tarefas específicas. Em jogos digitais, as RNAs podem ser utilizadas para controlar agentes, recebendo informações sobre o estado do jogo como entrada e produzindo ações como saída.

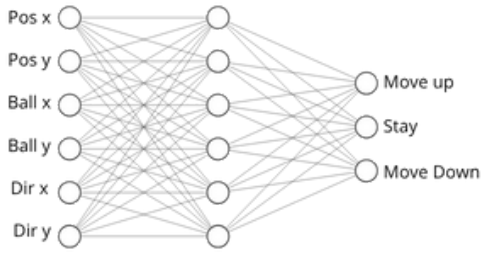

Figure 2.1: Exemplo de Rede Neural para um jogo

2.2 Algoritmos Genéticos (AGs)

AGs são algoritmos de busca e otimização inspirados nos princípios da seleção natural e da genética (Mitchell, 1998). Eles operam sobre uma população de indivíduos, onde cada indivíduo representa uma possível solução para o problema em questão. No contexto de treinamento de RNAs, cada indivíduo pode representar um conjunto de pesos e/ou a própria topologia da rede. O funcionamento de um AG pode ser resumido nas seguintes etapas:

- Inicialização: Uma população inicial de indivíduos é gerada aleatoriamente. No caso de treinamento de RNAs, isso significa criar uma população de redes neurais com pesos aleatórios e, no caso do NEAT, possivelmente topologias aleatórias também. A inicialização pode ser uniforme, onde todos os pesos são gerados com a mesma distribuição, ou pode ser mais sofisticada, levando em conta informações prévias sobre o problema.

- Avaliação: Cada indivíduo é avaliado de acordo com uma função de aptidão (fitness), que mede a qualidade da solução representada pelo indivíduo. No caso de jogos, a função de aptidão pode ser a pontuação obtida pelo agente controlado pela RNA, o tempo de sobrevivência, ou uma combinação de métricas relevantes. A escolha da função de aptidão é crucial para o sucesso do AG.

- Seleção: Indivíduos com maior aptidão têm maior probabilidade de serem selecionados para a próxima geração. Existem diversos métodos de seleção, como:

  - Seleção por torneio: Grupos de indivíduos são selecionados aleatoriamente, e o indivíduo com maior aptidão em cada grupo é escolhido para a próxima geração.
  - Seleção por roleta: Cada indivíduo tem uma probabilidade de ser selecionado proporcional à sua aptidão. Indivíduos com maior aptidão ocupam uma fatia maior da "roleta".
  - Seleção por ranking: Os indivíduos são ordenados por aptidão, e a probabilidade de seleção é baseada na posição no ranking, em vez do valor absoluto da aptidão.

- Cruzamento (Crossover): Novos indivíduos são gerados combinando partes dos indivíduos selecionados. O cruzamento simula a reprodução sexuada, permitindo a combinação de características benéficas de diferentes indivíduos. Existem diferentes métodos de crossover, como:

  - Crossover de um ponto: Um ponto de corte é escolhido aleatoriamente, e os genes antes do ponto de corte são herdados de um pai, enquanto os genes após o ponto de corte são herdados do outro pai.
  - Crossover de dois pontos: Dois pontos de corte são escolhidos, e os genes entre os pontos de corte são trocados entre os pais.
  - Crossover uniforme: Cada gene do novo indivíduo é herdado de um dos pais com igual probabilidade.

- Mutação: Pequenas alterações aleatórias são introduzidas nos novos indivíduos. A mutação introduz variabilidade na população, permitindo a exploração de novas áreas do espaço de busca e evitando a convergência prematura para soluções subótimas. A taxa de mutação, que define a probabilidade de um gene ser mutado, é um parâmetro importante do AG. Tipos de mutação incluem:
  - Mutação gaussiana: Um valor aleatório, tirado de uma distribuição normal, é somado ao valor do gene.
  - Mutação uniforme: O valor do gene é substituído por um valor aleatório dentro de um intervalo definido.
  - Bit flip mutation: Um bit específico é invertido.
- Repetição: As etapas de avaliação, seleção, cruzamento e mutação são repetidas por várias gerações, até que um critério de parada seja atingido, como um número máximo de gerações, um tempo limite, ou um nível de aptidão desejado.

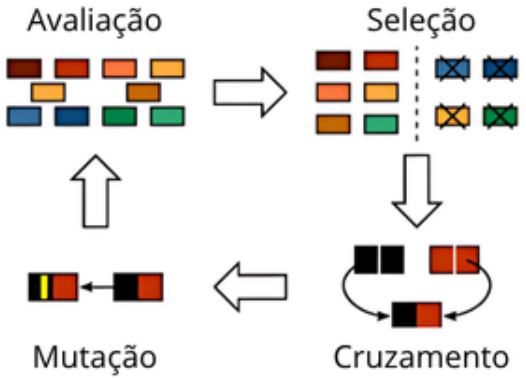

Figure 2.2: Loop do Algoritmo Genético

2.3 Neuroevolução e o Algoritmo NEAT

O algoritmo NEAT (NeuroEvolution of Augmenting Topologies), proposto por Kenneth O. Stanley em 2002, representa um avanço significativo na área de neuroevolução ao permitir que tanto os pesos quanto a própria topologia das redes neurais sejam evoluídos simultaneamente. Diferentemente dos algoritmos genéticos tradicionais aplicados a redes neurais, que normalmente operam sobre arquiteturas fixas e apenas otimizam os pesos das conexões, o NEAT é capaz de descobrir automaticamente arquiteturas de rede mais adequadas para o problema em questão, partindo de estruturas simples e aumentando gradualmente sua complexidade conforme necessário.

2.3.1 Princípios Fundamentais do NEAT

O NEAT se baseia em três princípios fundamentais:

- **Codificação Genética Inovadora:** Cada indivíduo na população do NEAT é representado por um genótipo que codifica não só os pesos das conexões, mas também

a estrutura da rede neural, incluindo os nós (neurônios) e as conexões entre eles. Para garantir a correta recombinação entre indivíduos com diferentes topologias, o NEAT utiliza *números de inovação*, que identificam de forma única cada gene de conexão criado durante a evolução. Isso permite alinhar corretamente os genes durante o cruzamento, preservando as inovações estruturais.

- **Complexificação Gradual:** Com o NEAT é possível começar a evolução com redes neurais mínimas. Ao longo das gerações, operadores de mutação podem adicionar novos nós ou conexões à rede, permitindo que a complexidade aumente apenas quando for benéfico para o desempenho. Essa abordagem evita o uso de arquiteturas superdimensionadas desde o início, promovendo soluções mais compactas e eficientes.

- **Especiação e Proteção da Diversidade:** Para evitar que inovações promissoras sejam rapidamente extintas devido à competição direta com indivíduos muito diferentes, o NEAT agrupa indivíduos semelhantes em espécies (*speciation*). A aptidão de cada indivíduo é ajustada de acordo com o tamanho de sua espécie, incentivando a exploração paralela de múltiplas estratégias evolutivas e protegendo novas estruturas até que possam ser refinadas.

2.3.2 Vantagens do NEAT

O NEAT apresenta diversas vantagens em relação a abordagens tradicionais de treinamento de redes neurais:

- **Descoberta Automática de Arquitetura:** Não é necessário definir previamente a arquitetura da rede, pois o próprio algoritmo descobre e adapta a topologia ideal ao longo da evolução.

- **Eficiência e Robustez:** A complexificação gradual evita o desperdício de recursos computacionais com redes desnecessariamente grandes, além de facilitar a generalização.

- **Preservação de Inovações:** O mecanismo de especiação protege novidades estruturais, permitindo que ideias inovadoras tenham tempo para amadurecer antes de competir com soluções estabelecidas.

- **Flexibilidade:** O NEAT pode ser aplicado a uma ampla variedade de problemas, especialmente aqueles onde a melhor arquitetura de rede não é conhecida a priori ou pode variar dinamicamente.

Chapter 3

Metodologia e Resultados

A metodologia empregada neste trabalho envolveu a implementação do algoritmo NEAT em Python, utilizando a biblioteca neat-python em conjunto com a biblioteca pygame. Quatro jogos foram selecionados como ambientes de teste para avaliar o comportamento do algoritmo em múltiplos cenários.

Durante o treinamento dos jogos, foram definidos alguns limitadores para que o processo não se tornasse desnecessariamente longo. Alguns exemplos de limitadores são: tempo máximo de jogo (visto que alguns jogos são repetitivos e basta uma amostra de tempo para aprender a jogar), pontuação máxima ou número de gerações.

3.1 Flappy Bird

Um jogo simples onde o agente controla um pássaro, devendo evitar colisões com canos. A entrada da rede neural consiste na altura do pássaro e na distância horizontal até o próximo cano. A saída é um valor binário que indica se o pássaro deve pular ou não. Devido à simplicidade do jogo, o NEAT convergiu rapidamente para uma solução eficaz, com o agente aprendendo a navegar pelos canos com sucesso.

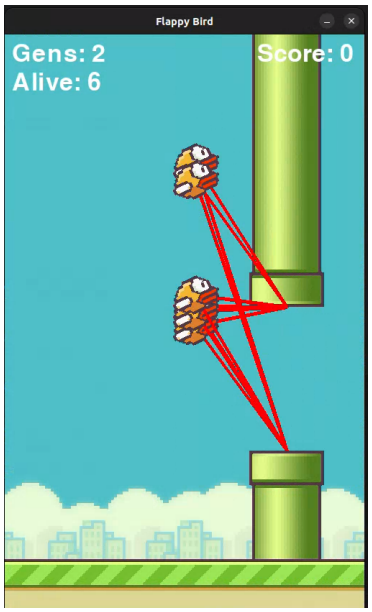

Figure 3.1: Rede Neural jogando jogo estilo Flappy Bird

Esse jogo foi escolhido como o primeiro ambiente de testes devido à sua simplicidade de implementação, assim servindo de propósito para entender as bibliotecas e o funcionamento do algoritmo. A convergência nesse cenário era tão rápida que foi necessário rodar o algoritmo com um número de indivíduos bem reduzido para que houvesse gerações antes que se chegasse no indivíduo ótimo.

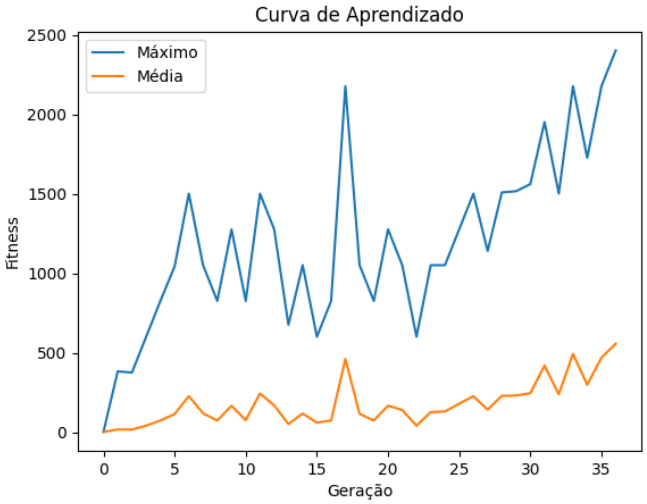

Figure 3.2: Gráfico da Aptidão por geração no jogo estilo Flappy Bird

3.2 Pong

Um jogo clássico em que o agente controla uma raquete, devendo rebater a bola para o lado do adversário, onde o oponente também possui uma raquete para rebater; quando um dos jogadores não consegue rebater, o outro pontua. A entrada da rede neural consiste na posição da raquete do jogador, na posição da bola e na direção em que a bola está indo. A saída consiste em três ações possíveis: subir, descer ou permanecer parado. O treinamento do agente para o Pong também obteve bons resultados, com o agente aprendendo a rebater a bola e a se posicionar estrategicamente.

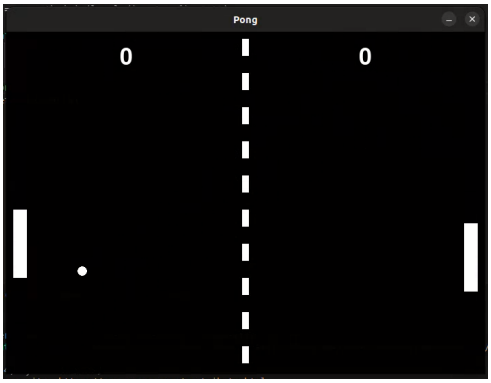

Figure 3.3: Rede Neural jogando jogo estilo Pong

Foi utilizada uma estratégia em que uma rede neural joga contra a outra. Essa estratégia não apresenta bons resultados no início, visto que ambos os jogadores ainda não

sabem jogar, mas, assim que em determinado ponto ambos já adquirem certo aprendizado, a velocidade de convergência aumenta drasticamente.

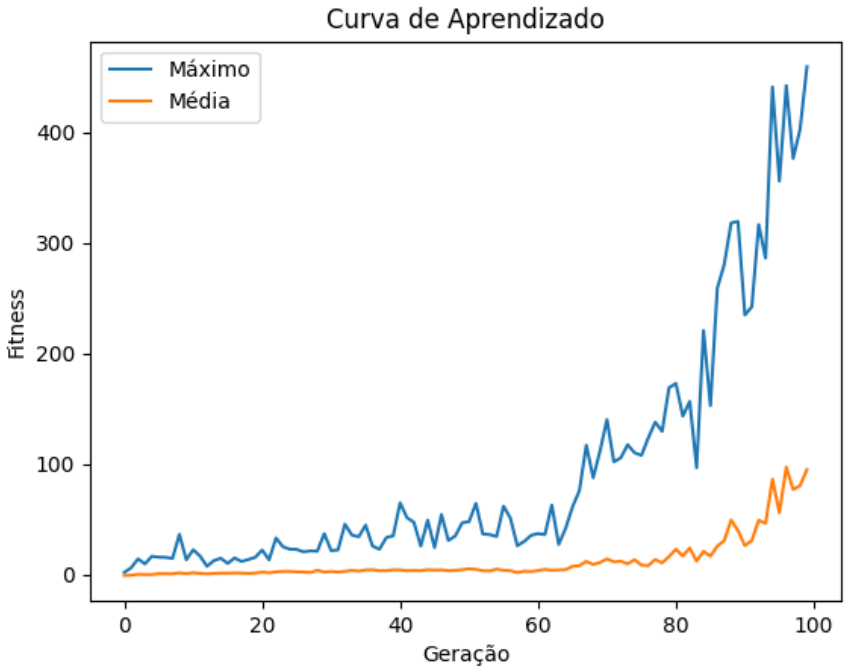

Figure 3.4: Gráfico da Aptidão por geração no jogo estilo Pong

3.3 Space Shooter

Um jogo baseado em um clássico, no qual o jogador controla uma nave e deve desviar (ou atirar) em meteoros que surgem na tela. Quanto mais tempo permanecer vivo e mais meteoros destruir, mais pontos o jogador acumula.

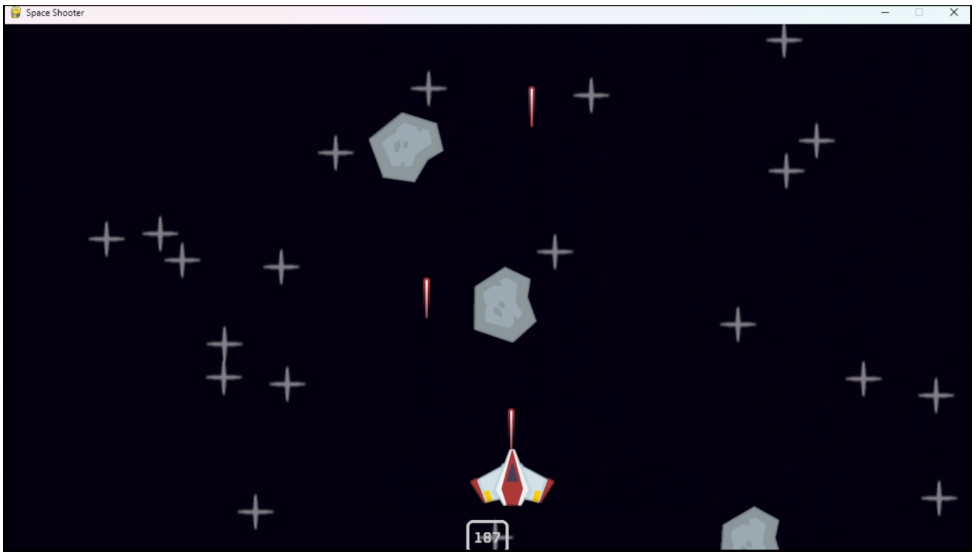

Figure 3.5: Rede Neural jogando jogo estilo Space Shooter

A entrada da rede neural consiste na posição da nave, na informação se o jogador pode atirar (ou está em cooldown) e em um tipo de radar, onde o espaço ao redor da nave foi dividido em 16 setores e, para cada um desses setores, é informada a distância até o meteoro mais próximo.

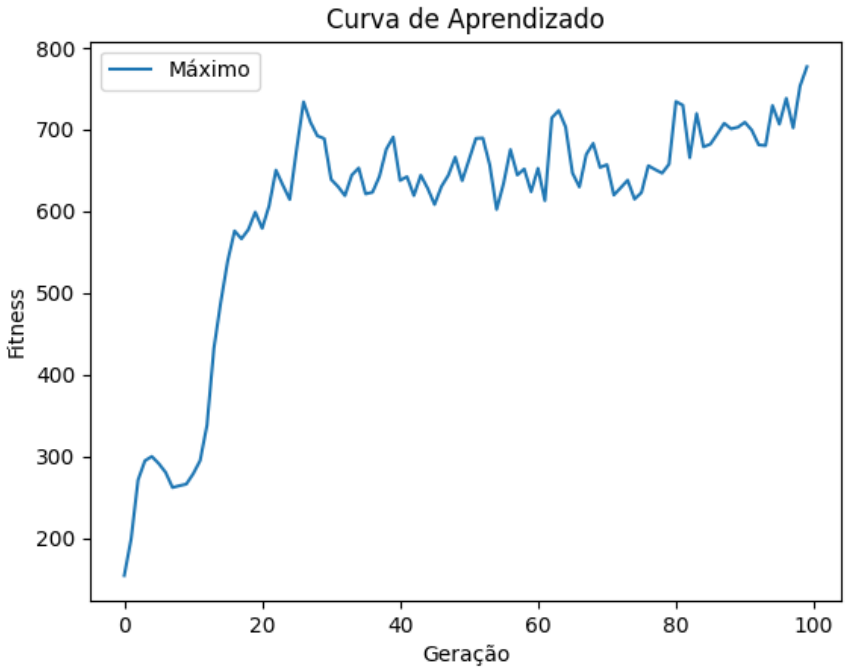

Figure 3.6: Gráfico da Aptidão por geração no jogo estilo Space Shooter

3.4 Snake

Um jogo em que o agente controla uma cobra, devendo comer alimentos para crescer de tamanho, enquanto evita colisões com as paredes e com o próprio corpo. A entrada da rede neural consiste em informações provenientes de 8 direções e na direção da cabeça. Para cada uma das oito direções, era informada a quantidade de blocos até a parede, a quantidade de blocos até a comida e a quantidade de blocos até o próprio corpo, sempre considerando a menor distância e normalizando os valores entre 0 e 1. A direção da cabeça poderia ser cima, baixo, esquerda ou direita, codificadas em one-hot encoding. A saída consiste em quatro ações possíveis: cima, baixo, direita e esquerda.

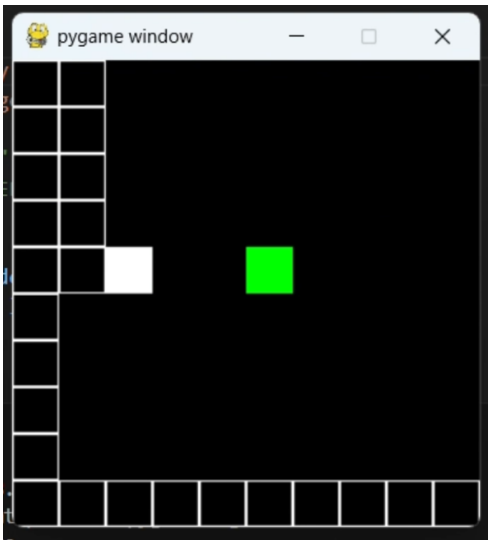

Figure 3.7: Rede Neural jogando jogo estilo Snake

O treinamento do agente para o Snake se mostrou significativamente mais desafiador. A complexidade do ambiente, a necessidade de planejamento a longo prazo e a dificuldade em representar o estado do jogo de forma eficiente contribuíram para essa dificuldade. Além disso, a definição da função de aptidão também foi um desafio, pois, ao mesmo tempo em que a quantidade de comidas era a informação mais importante, também era necessário penalizar indivíduos que andavam mais do que os outros (já que existem estratégias de ficar circulando por todos os blocos do jogo que não são ótimas), além de eliminar indivíduos que ficavam em loop, batiam nas paredes ou colidiam com o próprio corpo. Apesar da dificuldade, depois de muito treino foi possível encontrar indivíduos que completavam o jogo (fitness normalizado para 1)

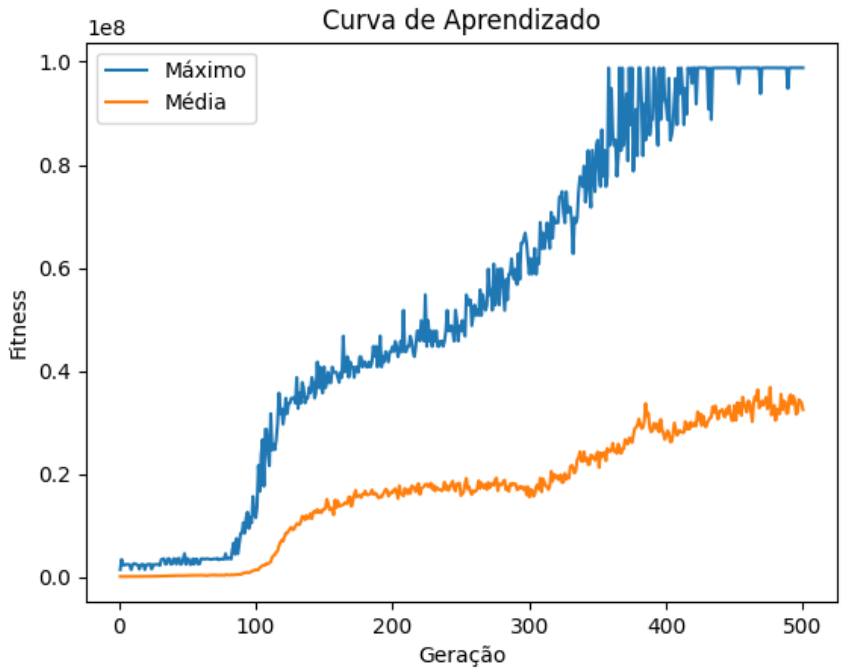

Figure 3.8: Gráfico da Aptidão por geração no jogo estilo Snake

Chapter 4

Conclusões

Os resultados obtidos demonstram o potencial do NEAT para treinar agentes em jogos digitais. A utilização do algoritmo genético como forma de otimização mostrou-se uma alternativa válida e interessante, pois sua capacidade de explorar novos espaços de busca e cruzar genomas são grandes aliadas na resolução de problemas de otimização.

O maior desafio encontrado é inerente a qualquer problema de otimização: encontrar uma função de aptidão que seja realmente significativa para obter o melhor resultado. É muito comum em estratégias de otimização que as redes "burlem" o processo, encontrando soluções que não são as ideais. Portanto, é necessário ajustar a função de otimização para direcionar os agentes no caminho desejado.

Gostaria de salientar que a oportunidade de estudar este assunto foi muito interessante e enriquecedora, dado que o cenário da inteligência artificial se encontra em grande expansão.

Chapter 5

Referências Bibliográficas

- Haykin, S. (2008). *Neural networks and learning machines*. Pearson Education.
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- CUNHA, A. G. *Manual de computação evolutiva e metaheurística*. Editora da Universidade Federal de Minas Gerais, 2013.
- Stanley, K. O. and Miikkulainen, R. (2002). *Efficient evolution of neural network topologies*. Evolutionary Computation
- Eiben, A. E., & Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer.
- Millington, I. (2006). *Artificial intelligence for games*. Morgan Kaufmann.