UNIVERSIDADE FEDERAL DO TOCANTINS
CÂMPUS UNIVERSITÁRIO DE PALMAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

INTELIGÊNCIA ARTIFICIAL BASEADA EM ALGORITMO GENÉTICO APLICADO NO JOGO SNAKE

BRUNO CAVALCANTI COSTA

PALMAS (TO)
2023

RESUMO

Neste estudo realizamos a implementação do famoso jogo Snake, assim como agente e modelo para treinamento de um algoritmo genético de forma a encontrar o melhor Score possível. Utilizamos PyGame para engine do jogo, PyTorch para o modelo entre outras bibliotecas para auxílio e otimização. Encontramos diversos problemas porém atingimos um resultado satisfatório assim como diversas melhorias possíveis.

Palavra-chave: LaTeX. UFTeX. Trabalho de Disciplina. Redação Científica.

SUMÁRIO

1 INTRODUÇÃO

Jogar videogames é desafiador e divertido; os jogos são considerados a área mais interessante de Inteligência Artificial (IA). Com a inteligência de simulação, os desenvolvedores de jogos estão treinando agentes e fornecendo jogos para ajudar a aprender muitos conceitos novos. Diferentes estratégias são seguidas, e linguagens de programação dinâmicas são usadas no desenvolvimento dos jogos. É utilizada uma combinação de redes neurais e Aprendizado Profundo para formular os jogos. Essa combinação oferece oportunidades para entender as coisas claramente e aprender o comportamento de reforço de forma eficaz.

Atualmente, uma combinação de Aprendizado de Máquina e Aprendizado por Reforço é utilizada no desenvolvimento de jogos. Os agentes são treinados e leituras são feitas para testar a eficiência do jogo. Certas regras são projetadas no jogo e os agentes do jogo são treinados para ajudar a resolver problemas específicos. No Aprendizado de Máquina, o Aprendizado por Reforço é uma das tecnologias mais intrigantes, na qual ações ótimas são aprendidas em qualquer estado por tentativa e erro. Esse processo de tentativa e erro ajuda a resolver problemas de forma eficaz e proporciona uma perspectiva única do aspecto de investigação de desempenho do Aprendizado por Reforço. (ALMALKI; WOCJAN, 2019)

Nessa pesquisa modificamos o jogo "Snake"de forma a avaliar a possibilidade de utilização de Algoritmo genético e aprendizado por reforço para completar o jogo da melhor forma possível.

Nos telefones celulares da Nokia, o jogo Snake foi o mais popular. O jogo Snake da Nokia é projetado para que um único jogador possa controlar o estado de funcionamento da cobra. Diferentes estágios são projetados no jogo para que a complexidade aumente gradualmente. Algumas regras devem ser seguidas no jogo. Por exemplo, a cobra tenta comer um item e corre atrás da comida, como resultado, pontos são acumulados. À medida que o tamanho da cobra aumenta, a complexidade do jogo também aumenta, tornando mais difícil obter uma pontuação alta.(ALMALKI; WOCJAN, 2019)

No nosso jogo alterado a cobra começa com tamanho 3 e uma "comida"aparece aleatorimente pelo mapa e a cobra cresce em 1 a cada "comida"coletada. A pontuação final depende unicamente da quantidade de "comida"coletada. O jogo termina caso a cobra tente fazer um movimento que a faria ocupar um espaço em que ela já ocupa ou quando encostaria na parede do mapa. Introduzimos Q-Learning através de PyTorch para os modelos de rede neural. O jogo foi feito utilizando PyGame que ajuda a gerenciar o código de forma a utilizar de agentes para assumir a função de usuário.

2 MÉTODOS

2.1 O jogo

Para realizar o treinamento foi feito o jogo com as seguintes regras:

- A cobra começa com tamanho 3
- O jogo acaba quando a cobra atinge a si mesma
- O jogo acaba quando a cobra atinge a parede
- O jogo acaba quando o limite de movimentos chega em 0
- A cobra começa com 150 movimentos
- A cada ”comida”coletada a cobra ganha 150 movimentos
- A cada ”comida”coletada o tamanho da cobra aumenta em 1

Temos uma classe ”Snake”que cria o display e realiza a integração do input do agente e a ação da cobra de forma que é possível visualizar o que a mesma está fazendo. Para isso utilizamos das bibliotecas PyGame (para exibir o jogo), Collections (para definir os pontos ao redor da cabeça), Numpy(para cálculos com arrays mais otimizados) e Enum(Para enumerar as direções que a cobra pode seguir).

2.2 Agente

Foi feito um agente que para tomar as decisões da cobra. Este agente recebe como input 11 variáveis, estas sendo:

- A posição da cabeça da cobra
- A posição da ”comida”
- Para cada ponto ao redor da cabeça da cobra (esquerda, direita e frente) coleta-se o estado ao redor desses pontos (perigo ou seguro, 1 ou 0).

O agente é responsável por passar o estado atual para o modelo e coletar o input a ser enviado para o jogo. Este estará em forma de um array entre:[1,0,0],[0,1,0] ou [0,0,1] dependendo da saída do modelo. O agente também é responsável por passar os parâmetros aos quais o modelo será treinado

2.3 Modelo

O modelo é onde o aprendizado acontece. O modelo foi criado com os 11 inputs anteriormente mencionados, 256 camadas ocultas e 3 camadas de saída. A função de ativação utilizada for a função de ativação linear retificada (Relu), a função fitness utilizada foi a equação de Bellman, a taxa de aprendizado foi 0.01 e o fator de desconto foi 0.9.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{j=0}^{T} \gamma^j r_{t+j+1}$$

Figura 1 – Equação de Bellman

3 RESULTADOS

Por fim, após treinamento, atingimos um score máximo de 72 em 250 gerações. Após análise do gráfico foi possível perceber um crescimento inconsistente na média da pontuação em algumas instâncias de treino a curva atingia formato quase linear enquanto em outras atingiu formato exponencial e transitava para logarítimico; Da mesma forma foi possível analisar o comportamento da cobra durante o treinamento e chegar a algumas observações:

- Foi possível observer que a cobra tende a evitar muito as bordas, muitas vezes entrando em loop quando a "comida"está encostada na parede
- A cobra tem uma tendência a se manter perto de seu próprio corpo, muitas vezes entrando em loop até o fim do jogo por conta disso
- Após algumas centenas de iterações o aprendizado é reduzido de forma que a pontuação estagna
- Por conta de sua proximidade com o próprio corpo a cobra frequentemente se colocava em situações inescapáveis

Por fim, gostaríamos de ressaltar possíveis melhoras:

- Essa implementação verificou apenas a vizinhança ao redor da cabeça da cobra e acreditamos que isso tenha causado os loops indefinidos assim como a condição de corredores inescapáveis, uma possível melhora para evitar isso seria coletar a distância da cabeça da cobra até as paredes e diferentes partes do corpo em múltiplas direções, aumentando a complexidade do input porém permitindo maior conhecimento da cobra quanto a seus arredores.
- Outra possível melhora seria mudar a funçào de fitness baseado no Score, priorizando tamanho da cobra em primeira instância e sobrevivência à partir de certo Score.
- Também seria possível introduzir "tipos"de comida diferentes de forma que a cobra tivesse que ativamente que evitar alguns tipos de comida. Isso também adicionaria maior compleixadade ao algoritmo.(ALMALKI; WOCJAN, 2019)

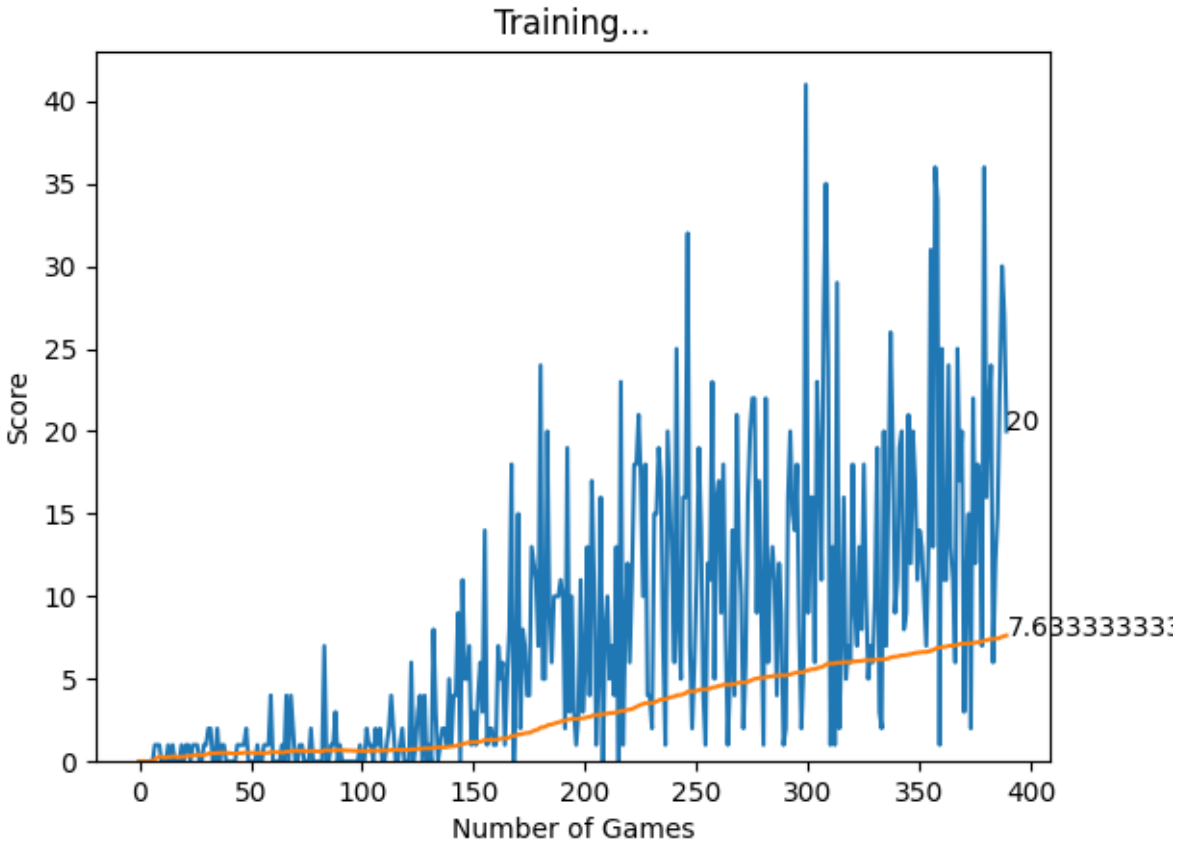

Figura 2 – Gráfico referente a uma dos treinamentos da cobra

REFERÊNCIAS

ALMALKI, A. J.; WOCJAN, P. Exploration of reinforcement learning to play snake game. In: IEEE. **2019 International Conference on Computational Science and Computational Intelligence (CSCI)**. [S.l.], 2019. p. 377–381.