Aplicação de uma rede perceptron no jogo Pong

Caio Varalta*
caio.varalta@unesp.br

Orlando Saraiva Júnior*
orlando.saraiva@unesp.br

***Resumo* — Este artigo explora a aplicação de uma rede perceptron em um clone do jogo clássico Pong. O foco do estudo é investigar as capacidades e limitações desta rede neural aplicada em um jogo digital simples. Os resultados dos experimentos indicam que a implementação do perceptron pode proporcionar uma experiência de jogo desafiadora e agradável, apesar das limitações encontradas.**

Palavras-chave* — *perceptron; pong; redes neurais; jogos digitais.

I. Introdução

No presente artigo, exploramos como uma rede perceptron pode ser aplicada em um clone do jogo clássico *Pong*. Logo no final da década de 1970, como explicado por Millington e Funge (2009, p.7, tradução nossa), "já havia clones do jogo *Pong* com raquetes controladas pelos oponentes (que basicamente seguiam a bola para cima e para baixo)"[1].

Portanto, a fim de explorar as possibilidades de aplicação de uma rede perceptron nos jogos digitais, e considerando a simplicidade do jogo *Pong*, que já tem implementações mais simples de IA desde a década de 1970, redigimos este artigo com base em duas perguntas principais:

1. *É possível desenvolver um bot (uma inteligência artificial) para o jogo Pong, com uma rede perceptron, que seja capaz de proporcionar uma experiência de jogo desafiadora e agradável aos jogadores?*
2. *É possível desenvolver um bot para o jogo Pong, utilizando a rede perceptron, que seja invencível?*

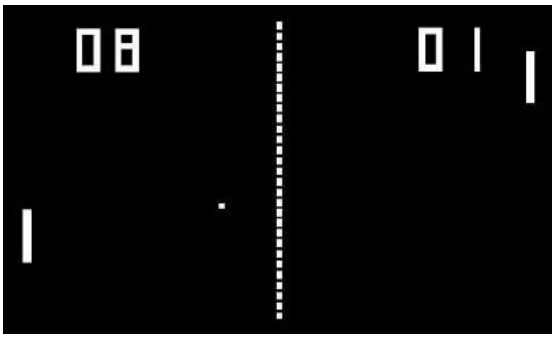
II. Revisão da Literatura

A. Pong

Em 1972, acordo com Kent (2001, p. 37, tradução nossa), "Nolan Bushnell, um engenheiro eletrônico da região norte da Califórnia, adaptou o brinquedo Magnavox de Ralph Baer — que permitia jogar ping-pong na tela da televisão — para ser jogado em uma máquina de moedas. Como o mundo já sabe, ele chamou isso de Pong."[2]

Segundo a explicação do autor, o Pong surgiu de um exercício. Pouco tempo depois de contratar o engenheiro Al Alcorn, Bushnell pediu que ele criasse um jogo que fosse simples de ser jogado: com uma bola, duas raquetes e uma pontuação — e mais nada na tela (KENT, 2001, p. 40).

Figura 1. Screenshot do jogo Pong (Atari).



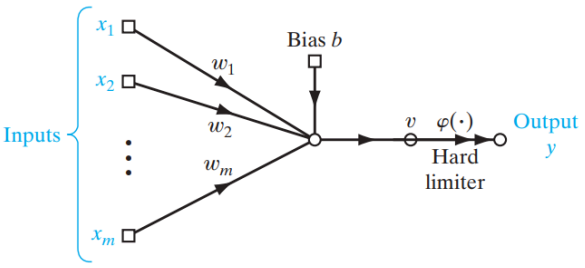
Fonte: Luz, 2014.

Após o lançamento, Pong se tornou um dos jogos que funcionavam com moedas mais lucrativos da história. As máquinas de Pong geravam (com frequência) um lucro semanal quatro vezes superior ao das outras máquinas (KENT, 2001, p. 53).

B. Perceptron

Como explicado por Haykin (2009, p. 47), o Perceptron ocupa um lugar especial no desenvolvimento histórico das redes neurais, pois ele foi a primeira rede neural descrita algoritmicamente.

Figura 2. Diagrama de fluxo de sinal do perceptron.



Fonte: Haykin, 2009.

A figura 2 apresenta o diagrama de fluxo de sinal do perceptron de Rosenblatt — que foi construído a partir do modelo de neurônio de McCulloch-Pitts, como Haykin pontua na mesma obra.

O autor explica que esta modelagem neural consiste em um combinador linear seguido por um limitador rígido (*Hard limiter*). O nó somatório do modelo neural realiza o cálculo da seguinte maneira:

* Ambos os autores contribuíram igualmente com a produção do artigo.
[1] No original: *Up to that point there had been Pong clones with opponent-controlled bats (that basically followed the ball up and down) (...).*
[2] No original: *Nolan Bushnell, a rather clever electronics engineer from Northern California, adapted Ralph Baer's Magnavox toy for playing ping-pong on the television screen into a coin machine. As the world knows, he called it Pong.*

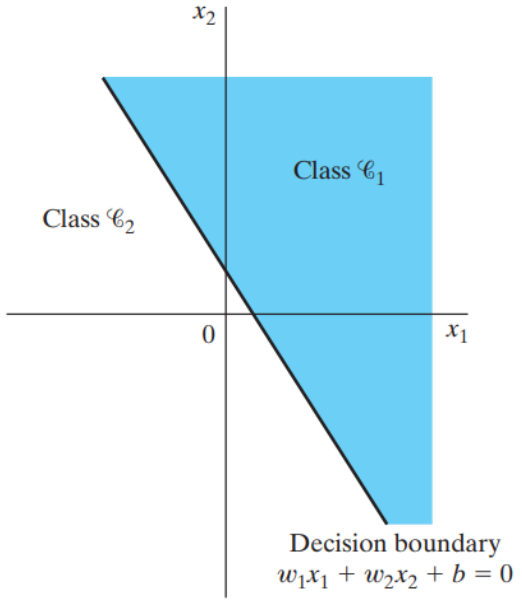
$$v = \sum_{i=1}^{m} w_i x_i + b$$

Onde calcula-se uma combinação linear das entradas (x_1, x_2,..., x_m) aplicadas aos seus respectivos pesos sinápticos (w_1, w_2,..., w_m), bem como incorpora um viés externamente aplicado — representado na figura 2 como *Bias*, ou *b*.

E então, a soma resultante é aplicada ao limitador rígido (φ). Como o limitador aplica uma função sinal, o neurônio produz uma saída igual a 1 se a entrada é positiva e -1 se for negativa.

Através desse fluxo matemático, o perceptron é capaz de classificar padrões que são linearmente separáveis. Segundo Haykin (2009, p. 48), o perceptron é reconhecido como a forma mais simplificada de rede neural usada para a classificação de padrões considerados linearmente separáveis.

Figura 3. Ilustração do hiperplano como limite de decisão para um problema de classificação de padrões bidimensional e com duas classes.



Fonte: Haykin, 2009.

Contudo, o autor ressalta que as classes têm de ser linearmente separáveis para o perceptron funcionar corretamente. Este é um ponto relevante a ser considerado em suas implementações e foi o ponto que nos levou às questões apresentadas na introdução deste artigo.

III. Implementação do Perceptron em Python

Nesta seção, descreveremos as etapas da construção do bot em Python, onde os códigos encontram-se disponíveis no repositório do *GitHub* criado para o artigo (SARAIVA JR, 2024). Todas as versões do jogo Pong criadas enquanto realizamos nossa pesquisa e testes utilizam a biblioteca *PyGame*.

Como descrito na página oficial da PyGame (2024), a PyGame é uma biblioteca em Python utilizada para a criação de jogos e outras aplicações multimídia. Ela fornece funcionalidades para manipulação de gráficos, sons e interações com o usuário, facilitando o desenvolvimento de jogos 2D. A PyGame é baseada na biblioteca SDL (*Simple DirectMedia Layer*), que é uma biblioteca em C utilizada para acesso a hardware de baixo nível.

C. Pong versão 1 (pong_v1.py)

Em nossa primeira implementação do Pong, nosso foco foi entender o mecanismo do jogo e explorar a biblioteca PyGame. Inicialmente, para separar as responsabilidades internas do jogo, foi criada uma classe para representar a bola (representada no código pela *class Ball*) e uma classe para representar as paletas, ou raquetes (representadas no código pela *class Paddle*).

No jogo existem três objetos com responsabilidades distintas: objeto *ball*, que representa a bola do jogo, instanciado na linha 58 do código; o objeto *player1*, que representa o primeiro jogador (instanciado na linha 59); e o objeto *player2*, que representa o segundo jogador e oponente do *player1* (instanciado na linha 60).

A sessão do jogo ocorre dentro da função main (linhas 53-131), especificamente em um loop infinito (linhas 67-129), com a condição de saída sendo o instante em que o contador de pontos de um dos jogadores for igual a cinco.

Nesta versão, dois jogadores humanos devem interagir: o jogador 1 utiliza as teclas "W" e "S" para movimentar sua raquete para cima e para baixo, respectivamente, e o jogador 2 utiliza as teclas direcionais "cima" e "baixo" para movimentar a sua raquete para cima e para baixo.

D. Pong versão 2 (pong_v2.py)

Em nossa segunda implementação de Pong, nosso interesse focou-se em criar um bot para o jogador 1 que fosse invencível. Para isso, a solução desenvolvida foi a seguinte: o jogador 1 (bot) movimenta sua raquete no eixo x na mesma posição do eixo x da bola.

O jogador 2 é o humano que estará jogando contra o bot, e ele utiliza as teclas direcionais "cima" e "baixo" para movimentar sua raquete para cima e para baixo.

Nesta versão, o jogador humano não possui a possibilidade de vencer, pois o bot sempre terá sua raquete alinhada com a bola. Assim como a primeira implementação, esta versão ainda não faz uso da rede Perceptron.

E. Pong versão 3 (pong_v3.py)

Na terceira implementação, o nosso foco deu-se na coleta de dados para a futura criação de uma rede Perceptron. Nesta implementação, a função *register_in_dataframe* (definida nas linhas 20-39), armazena os dados da bola: coordenada x, coordenada y, velocidade x, velocidade da bola em y, e a ação a ser tomada ("*UP*" ou "*Down*").

Na linha 13, define-se a variável de *samples*. Ela representa a quantidade de iterações que será executada na função *register_in_dataframe*. Após a quantidade definida de iterações serem realizadas (150 vezes, na versão do código presente no GitHub), gera-se os arquivos features.pkl e targets.pkl

F. Leitura dos dados gerados (reading_the_data.py)

O código reading_the_data.py é o primeiro código que faz uso da biblioteca scikit-learn, uma das mais populares bibliotecas de aprendizado de máquina em Python (SCIKIT-LEARN, 2024). A biblioteca fornece uma ampla gama de ferramentas para modelagem de dados, algoritmos de classificação, regressão, clustering, redução de dimensionalidade, pré-processamento de dados e validação de modelos.

O código em questão recebe os arquivos features.pkl e targets.pkl como insumos, e utiliza um modelo de aprendizado de máquina (Perceptron) para treinar e testar várias vezes, com diferentes estados aleatórios, a fim de avaliar a precisão do modelo.

O desenvolvimento deste programa ajudou a avaliar a robustez e a variabilidade da precisão do modelo em diferentes cenários e nos permitiu escolher o melhor número aleatório (para utilizar como *Random State*) baseado nos dataframes (arquivos .pkl) gerados na versão 3 do Pong.

G. Pong versão 4 (pong_v4.py)

Na quarta implementação do jogo Pong, o jogador 1 é controlado pelo bot invencível desenvolvido na versão 2, enquanto o jogador 2 é um bot controlado pela rede Perceptron (definido na linha 17 do código), treinada a partir dos dataframes gerados na versão 3, em conjunto com o código de leitura dos dados (*reading_the_data.py*).

O jogo inicia e continua até que um dos jogadores atinja a pontuação máxima (definida nas linhas 128 e 134), momento em que o programa exibe a mensagem de vitória e espera por 2 segundos antes de encerrar.

A escolha do número aleatório (*Random State*) mais adequado para o Perceptron foi possível após análise dos resultados apresentados pelo script reading_the_data.py. Através dos nossos experimentos, e baseado nos dataframes .pkl gerados, o número 43 foi o mais plausível que encontramos para atingirmos um resultado satisfatório com a rede Perceptron.

H. Pong versão 5 (pong_v5.py)

Na quinta (e última) implementação, o jogador 1 é controlado por um humano, através das setas direcionais “cima” e “baixo”, e o jogador 2 é um bot controlado pela rede Perceptron, com as mesmas configurações apresentadas na versão 4.

Essa versão foi criada para averiguar a dificuldade do bot contra o jogador humano, e analisar como o comportamento final do bot se encaixa no contexto das perguntas que guiaram nossa pesquisa, apresentadas na introdução do artigo.

IV. Experimentos e Resultados

Realizamos diversos testes contra o bot na versão 5 do Pong desenvolvida. Inclusive testamos realizando alterações nos parâmetros do jogo sem retreinar o perceptron — como a redução da velocidade da bola (alterando *ball_speed* para 6, na linha 9 do código), e do tamanho da bola (alterando *self.size* para 10, na linha 29 do código). Estas alterações de parâmetro trouxeram uma performance melhor para a rede Perceptron treinada.

Um ponto de destaque é que, após as alterações realizadas, o valor máximo de rebatidas que a rede Perceptron conseguiu realizar foi de 3 vezes na sequência (para o mesmo ponto dentro do jogo).

Na última partida realizada contra o perceptron, ele conseguiu rebater a bola um total de 9 vezes dentro de 8 pontos marcados (sendo 5 pontos realizados pelo jogador humano, e 3 pelo Perceptron).

Nesta partida em questão, por três pontos seguidos, o Perceptron não conseguiu rebater a bola, e nos três pontos na sequência, ele rebateu a bola 2 ou 3 vezes seguidas.

Ao comparar as escolhas de movimento do Perceptron neste experimento realizado com as escolhas de movimento realizadas pelo bot no jogo original, como observadas através do vídeo disponível pelo canal do youtube Old Classic Retro Gaming (2013), chegamos à conclusão de que a rede Perceptron apresentou escolhas de movimentos melhores e mais “inteligentes” do que a versão original do jogo, de 1972.

Contudo, baseado nos resultados que obtivemos através dos testes com o bot, também notamos que a rede Perceptron desenvolvida para o experimento ofereceu um desafio limitado. Ela não realiza suas escolhas de movimento de forma aleatória ou imprecisa, mas também não oferece um desafio para jogadores com mais experiência no Pong (como os jogadores que conseguem rebater a bola 4 vezes seguidas em todos os *rounds*).

V. Discussão e Conclusão

Baseado nos resultados dos experimentos, pode-se argumentar que é possível desenvolver um bot para o jogo Pong, com uma rede Perceptron, que seja capaz de proporcionar uma experiência de jogo desafiadora e agradável aos jogadores.

Ela é uma experiência agradável pois o comportamento do Perceptron é semelhante ao comportamento do jogador: ele tenta seguir a bola assim que os rebotes acontecem e às vezes reconsidera sua movimentação e se mantém no local para acertar a bola.

Já com relação ao desafio, fica-se o questionamento do nível de desafio que deseja-se atingir com a rede Perceptron. Nos nossos experimentos, o bot é desafiador até um certo limite: como contra jogadores que vão rebater a bola de 1 a 3 vezes por *round*. Para jogadores experientes, a experiência não será desafiadora.

Portanto, uma das principais recomendações para trabalhos futuros, que queiram explorar e expandir os códigos desenvolvidos para o presente artigo, seria a tentativa de

conseguir gerar *Random States* para o Perceptron que tivessem uma acurácia maior do que a utilizada no experimento — sendo que nosso melhor *Random State* foi o de número 43 com, aproximadamente, 0.80 de acurácia, segundo o arquivo *reading_the_data.py*.

Outra recomendação seria a de explorar a utilização de outras estruturas de redes neurais para o treinamento da IA. Como o Perceptron, sendo a mais simples delas, conseguiu trazer resultados satisfatórios para a IA do jogo, redes neurais com maior capacidade de classificações de padrões certamente trarão resultados melhores para o experimento.

Por fim, como nosso melhor *Random State* apresentou acurácia de 0.80, não conseguimos criar um bot Perceptron que fosse invencível para este experimento do jogo Pong. Ainda existe espaço para a exploração do presente projeto e código, então não argumentaremos que seja impossível, mas não conseguimos validar ou invalidar completamente esta pergunta, que nos guiou durante nossos experimentos.

Também concluímos, através dos experimentos, que a utilização de um bot com uma rede Perceptron pode melhorar significativamente a experiência do jogador em jogos como o Pong.

Um bot com Perceptron pode ser projetado para oferecer um nível de desafio que é justo, mas não impossível de superar. Isso pode aumentar a satisfação do jogador ao vencer uma máquina que está tentando "ganhar" de forma justa, em vez de, simplesmente, ser invencível.

Integrar um bot com uma rede Perceptron em jogos como Pong podem trazer uma experiência mais equilibrada, realista e envolvente para o jogador humano.

Referências

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. [s.l.] Pearson Prentice Hall, 2009.

KENT, S. L. **The ultimate history of video games: from Pong to Pokemon - the story behind the craze that touched our lives and changed the world**. 1. ed. Nova Iorque, Nova Iorque: Three Rivers Press, 2001.

LUZ, A. R. D. Evolution of the Physical Interfaces in Videogames as a Support to the Narrative and the Gaming Experience. Em: MARCUS, A. (Ed.). **Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments**. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014. v. 8518p. 688–698.

MILLINGTON, I.; FUNGE, J. **Artificial Intelligence for Games**. 2. ed. Boca Raton: CRC Press, 2009.

OLD CLASSIC RETRO GAMING. **Arcade Game: Pong (1972 Atari) [Re-Uploaded]**. Disponível em: https://www.youtube.com/watch?v=e4VRgY3tkh0. Acesso em: 26 jun. 2024.

PYGAME. ***About — Wiki***. Disponível em: https://www.pygame.org/wiki/about. Acesso em: 25 jun. 2024.

SARAIVA JR, Orlando. **Pong**. Disponível em: https://github.com/orlandosaraivajr/Pong . Acesso em: 25 jun. 2024.

SCIKIT-LEARN. ***Scikit-learn: Machine Learning in Python***. Disponível em: https://scikit-learn.org/stable/. Acesso em: 25 jun. 2024.