CRIAÇÃO DE CONHECIMENTO ATRAVÉS DA PESQUISA E IMPLEMENTAÇÃO DA REDE NEURAL PERCEPTRON E SUAS APLICAÇÕES COM PORTAS LÓGICAS

PEDRO HENRIQUE DIEHL[1]; LÍDIA MARTINELLI DE OLIVEIRA[2]; LUCAS MAYDANA MENDES[3]; LUCAS PEREIRA[4]; MIGUEL BECK BERNO[5]; ELMER ALEXIS GAMBOA PEÑALOZA[6]

[1]*Universidade Federal de Pelotas – diehl.pedroh@gmail.com*
[2]*Universidade Federal de Pelotas – lidiamartinelli99@gmail.com*
[3]*Universidade Federal de Pelotas – lucas.maydana@gmail.com*
[4]*Universidade Federal de Pelotas – lucaspereiraifsul12@gmail.com*
[5]*Universidade Federal de Pelotas – miguel.bberno@gmail.com*
[6]*Universidade Federal de Pelotas – eagpenaloza@ufpel.edu.br*

1. INTRODUÇÃO

De acordo com MCCULLOCH; WARREN S. e WALTER PITTS, Bulletin of mathematical biology, 1943), que propuseram que neurônios são capazes de implementar operações lógicas através de redes neurais computacionais. A partir do trabalho desenvolvido por McCulloch e Pitts, Frank Rosenblatt(1928-1971), desenvolveu o modelo inicial do Perceptron no ano de 1958.

O Perceptron é um algoritmo de estrutura simples, utilizado para separar padrões linearmente separáveis, reconhecendo-os baseados em uma rede neural que utiliza duas camadas simples de adição e subtração (Haykin, 1999; Bishop, 2008). Neste trabalho são apresentados a metodologia e resultados práticos de simulação do processo de ensino desenvolvido na disciplina de Matemática Discreta através da aplicação do algoritmo Perceptron em portas lógicas.

2. METODOLOGIA

A motivação para o projeto iniciou com o desafio de pesquisar, avaliar e implementar o Perceptron como técnica básica para fundamentos de inteligência artificial. O desenvolvimento do artigo foi dividido nos seguintes itens: pesquisa realizada, desenvolvimento do modelo, simulação e validação dos resultados.

2.1. INTRODUÇÃO A ARQUITETURA DO PERCEPTRON

A rede neural Perceptron consiste em uma camada única de *x* neurônios, conectada às entradas por pesos $w_{j,k}$ (Os índices *j* e *k* são a força de conexão do *j-ésimo* neurônio com a *k-ésima* entrada). Estes neurônios são somados, subtraindo-se o limiar (*bias*), então uma função de ativação *f(t)* é aplicada e por fim, uma saída é compilada.

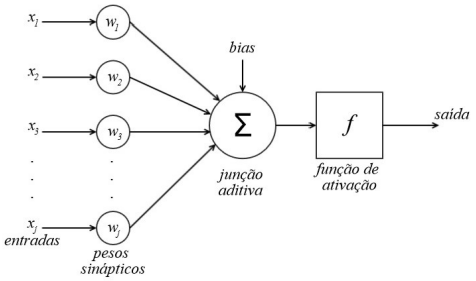


Figura 1: Arquitetura do Perceptron

2.2. DESENVOLVIMENTO DO MODELO

Com intuito de facilitar a compreensão das deduções matemáticas posteriores, uma tabela para consulta Tab. (1) foi anexada:

Variável	Legenda
w_i	Peso
θ	Limiar / Bias
x^k	Vetor da k-ésima amostra
d^k	Saída desej. [k-ésima amostra]
y	Saída do Perceptron
η	Taxa de aprendizado

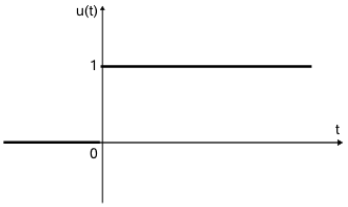
Tabela 1: Tabela de consulta para significado dos símbolos

Visando obter o potencial de ativação bem como o seu resultado após a compilação pela função de ativação Fig. (3) que corresponde à soma dos valores de entrada multiplicados pelos seus respectivos pesos obtendo uma saída linear do neurônio, extraímos o sistema de equações presentes em Fig. (2).

$$\begin{cases} \mathrm{u} = \sum_{i=1}^{n} w_i x_i - \theta \rightarrow w_1 x_1 - \theta + ... + w_n x_n - \theta \\ y = g(u) \end{cases}$$

Figura 2: Dedução matemática inicial

Funções de ativação geralmente utilizadas em modelos de Perceptrons, possuem intervalos entre [-1, 1] ou [0, 1], a qual foi a escolhida e apresenta dedução na Fig. (3).



$$u(t) = \begin{cases} 1, \ se \ t > 0 \rightarrow \mathrm{x}_n \ \in Classe \ A \\ 0, \ se \ t < 0 \rightarrow \mathrm{x}_n \ \in Classe \ B \end{cases}$$

Figura 3: Função de ativação (Degrau unitário) com intervalo de [0, 1]

O sistema de equações para treinamento do neurônio foi deduzido matematicamente em Fig. (4), este garante o condicionamento correto para tomada de decisões ideal por parte do modelo.

$$\begin{cases} \mathrm{w}_i^{atual} = w_i^{anterior} + \eta \cdot (d^k - y) \cdot x_i^k \\ \theta^{atual} = \theta^{anterior} + \eta \cdot (d^k - y) \cdot (-1) \end{cases}$$

Figura 4: Sistema de equações para treinamento do neurônio artificial

2.3. ESPECIFICAÇÕES DE HARDWARE

A simulação foi desenvolvida em utilizando um notebook com as seguintes características operacionais:

Modelo Inspiron 5584 com processador Intel® Core™ i7-8565U CPU @ 1.80GHz 1.99GHz, RAM de 8,00GB e sistema operacional de 64 bits, Windows 10 e processador baseado em x64. Para atingir melhor processamento de dados, utilizou-se de um SSD 970 EVO Plus da Samsung® em conjunto com a placa de vídeo NVIDIA® GeForce MX130.

2.4. ADEQUAÇÕES E SIMULAÇÃO

Devido ao fato do sistema dispor de diversas entradas, a abordagem vetorial é a mais adequada. Para simular o Perceptron, foi desenvolvido em MATLAB® os algoritmos presentes em Alg. (1 e 2), foram realizados testes para garantir um ajuste fino visando encontrar os valores ideias para os peso(s) e limiar(es) utilizados, pois são estes que garantem resultados corretos de forma eficaz.

O funcionamento em conjunto dos algoritmos descritos abaixo acontece de maneira na qual, a cada iteração, o par de entradas é comparado com sua saída, gerando um potencial de ativação que é aplicado à função presente em Fig. (3), a partir do resultado obtido é calculado o erro que é utilizado para ajustar os valores do(s) peso(s) e limiar(es).

Algoritmo 1: Algoritmo para executar o Perceptron no MATLAB®

```
ENTRADA   ▷ Vetor bidimensional com pares de entradas da porta lógica
ALVO      ▷ Vetor contendo a saída equivalente ao par da variável entrada
peso := 1                          ▷ Definido pesos iniciais para as entradas
bias := 1                          ▷ Definindo o limiar de aprendizado inicial
epoca := 1                         ▷ Inicializando o contador de iterações
para j := 1 : epoca faça
    para i := 1 : size(ENTRADA, 2) faça
        vetor := ENTRADA                    ▷ Seleciona um par de entradas
        alvo := ALVO_i          ▷ Seleciona a saída equivalente a essas entradas
        peso, bias := perceptron(vetor', alvo, peso, bias)
    fim para
fim para
```

Algoritmo 1: Código de execução do Perceptron no MATLAB®

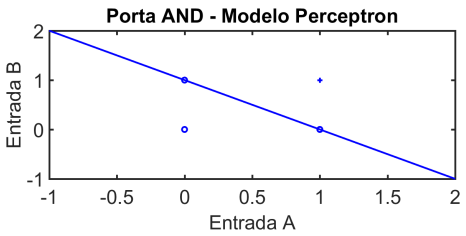
Algoritmo 2: Função para Perceptron no MATLAB®

```
perceptron (vetor: a1, a2, 0 ou 1; alvo: 0 ou 1, inteiros; peso: a1, a2, flutuantes;
bias: flutuantes)
  funcao_ativacao := hardlim(peso * vetor' + bias)
  erro := alvo − funcao_ativacao
  peso_novo := peso + erro * vetor
  bias_novo := bias + erro
retorna peso_novo, bias_novo   ▷ Devolve os valores atualizados para novas
                                 tomadas de decisão do neurônio artificial
```

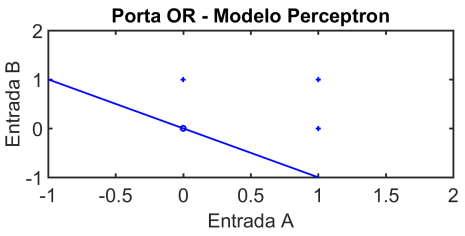
Algoritmo 2: Função para Perceptron no MATLAB®

3. RESULTADOS E DISCUSSÃO

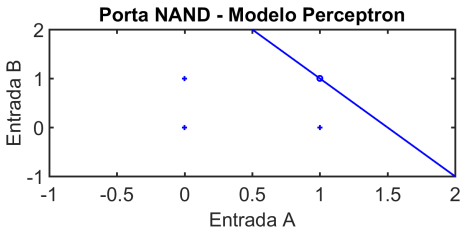
Os resultados obtidos da implementação dos algoritmos expostos na metodologia estão ilustrados nas Figs. (5). Pode-se observar a eficiência do algoritmo para encontrar os resultados lógicos das portas lógicas testadas, ex: AND, OR, NAND e NOR. De outro lado, observa-se que para portas lógicas ou funções booleanas compostas, os algoritmos estudados não atingem o resultado ideal, um exemplo disso é a comporta XOR.



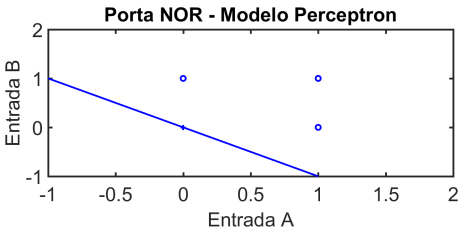
a)



b)



c)



d)

Figura 5: Resultados para as portas lógicas: a) AND ; b) OR; c) NAND e d) NOR. Obtidos com o Algoritmo Perceptron desenvolvido.

4. CONCLUSÕES

A análise do Perceptron, utilizando a função de degrau unitário, comprovou-se veloz e efetiva, ao passo que com poucas épocas já conseguiu identificar e rotular corretamente os modelos de portas lógicas que podem ser rotulados de forma linear. Todavia, o neurônio artificial não conseguiu categorizar corretamente os modelos que apresentam outras configurações de disposição no plano ou exigem outras camadas de decisões lógicas. Para esses casos o ideal seria utilizar a rede neural Multilayer Perceptron, que permite ter mais de uma saída e número indeterminado de neurônios em conjunto com outra função de ativação, possibilitando gerar resultados precisos de forma ágil e eficiente.

É importante concluir que o processo de ensino e aprendizado sobre algoritmos básicos de inteligência artificial através dos exemplos práticos de pesquisa e aplicação, motiva os estudantes a aprenderem através de buscas autodidatas e assim, desenvolvendo bases sólidas de conhecimento para aplicação em futuras disciplinas e na vida profissional.

5. REFERÊNCIAS BIBLIOGRÁFICAS

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **Bulletin of mathematical biology**, v. 52, n. 1, p. 99-115, 1990.

KUBAT, Miroslav. Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7. **The Knowledge Engineering Review**, v. 13, n. 4, p. 409-412, 1999.

BURR, Thomas et al. Pattern Recognition and Machine Learning. Christopher M. Bishop. **Journal of the American Statistical Association**, v. 103, p. 886-887, 2008.

RABELO, Rafael Tolentino. Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado à robótica móvel. 2014.