Trabalho 1: Perceptron

Pedro Victor Dos Santos Matias, 21601225

pvsm@icomp.ufam.edu.br

I. Introdução

O presente trabalho é sobre um problema ilustrativo com o desenvolvimento e treinamento de classificadores neurais, mais precisamente aplicada a uma abordagem de classificação de peças. A arquitetura da rede projetada é baseada no modelo Perceptron de Rosenblatt (1958) e foi programada em MATLAB (Matrix Laboratory).

O contexto do problema apresenta as medições de 4 conjuntos de peças (Cp1,Cp2,Cp3 e Cp4) em torno de duas características. Na primeira parte do experimento as peças irão passar em uma esteira e um robô irá separar as peças do conjunto 1,2 e 3 em uma cesta, acionado com 1 lógico, e a 4 em outra, com o 0 lógico. Já na segunda parte teremos que especificar e treinar outros classificadores a partir de outra lógica de discriminação dos padrões.

Este documento está organizado em 6 seções. Na segunda, será realizado uma revisão bibliográfica sobre o modelo Perceptron . No capítulo 3 é descrita a arquitetura geral da rede e o algoritmo de treinamento para solução do problema. Os resultados, como gráficos e modelos são apresentados na seção 4. Na seção de conclusão é realizado uma comparação entre os classificadores obtidos. E por último um apêndice com a listagem dos códigos em MATLAB.

II. Revisão Bibliográfica

No próximo ano, o Perceptron estará fazendo 50 anos, um dos algoritmos mais tradicionais para aprendizagem supervisionada. Inventado em 1958 e concluído em 1960 pelo cientista Frank Rosenblatt, e se encaixa no paradigma da Inteligência Artificial Conexionista. Esse segmento refere-se inteligência como produto da implementação de um arquitetura semelhante a estrutural biológica do cérebro a partir de modelos matemáticos.

O Perceptron é um modelo matemático que recebe um conjunto de entradas $[x_1, x_2, ..., x_n]$ e produz uma única saída binária. Rosenblatt, para calcular a saída, propôs a introdução de um conjunto de pesos $[w_1, w_2 e w_n]$, que são números reais expressando a importância de respectivas entradas para a saída. Essa saída binária é determinada pela combinação linear dos pesos pela entrada e a polarização, delimitada por um valor de limiar (*threshold*).

Apesar de ser a arquitetura mais simples de uma rede neural, podemos a considerar como um grande passo no avanço dos estudos em modelos de aprendizagem. Além disso, ele permite uma compreensão clara de como funciona uma rede neural em termos matemáticos e apesar de não ser o modelo de neurônio artificial mais comum atualmente, é considerado uma excelente introdução no campo. Em uma publicação de Minsky and Papert (1969) prova que a rede Perceptron era limitada a resolução de problemas linearmente separáveis, como as operações lógicas OR e AND (Johnson (1993)). Esse fato levou um atraso na pesquisa de sistemas conexionistas o que posteriormente foi resolvido com aplicação de multicamadas.

Permitindo assim o surgimento de novas formas de aprendizagem e de arquiteturas mais robustas.

Atualmente há o conceito de multicamada de Perceptrons e técnicas de aprendizagem mais complexas e otimizadas para determinados problemas de reconhecimento de padrões e classificação. Como no caso apresentado em Guha and Patra (2010) com a aplicação no problema de interferência co-canal em redes sem fio, que são pertubações causadas na comunicação devido o reuso de frequência em estações distribuídas equitativamente em diferentes clusters.

III. Metodologia

Nesta seção iremos informar quais os métodos foram utilizados para realizar este estudo, qual o instrumento usado o processamento da rede, especificação da arquitetura de acordo com o cenário do problema e o algoritmo-solução.

Classifica-se este estudo como explicativo pois relaciona a teoria apresentada em sala de aula referente ao modelo simples de rede Perceptron (aprendizagem supervisionada) com um problema prático de classificação. Do ponto de vista dos procedimentos, o trabalho em mãos faz a opção por métodos experimentais. Como a representação de um neurônio artificial é realizada por uma estrutura matemática, utilizou da codificação de algoritmos para treinamento e processamento da rede com o software e linguagem de programação MATLAB.

I. Parte 1

Primeiramente o problema apresenta um gráfico com 4 conjuntos de peças, relacionados com o um par de características (P(1) e P(2)), como descrito na figura abaixo.

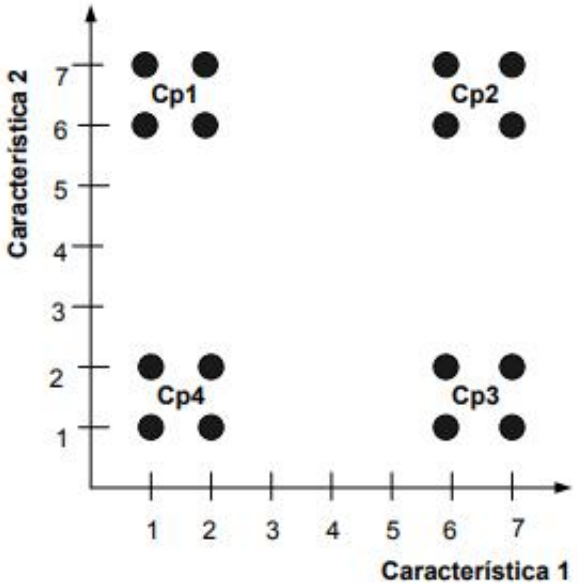


Figura 1: Conjunto de peças

No primeiro caso do problema o robô é acionado com nível lógico 1 quando as peças

do conjunto Cp1,Cp2 e Cp3 passam pela esteira e são recolocadas na cesta 1, que forma um agrupamento (*cluster*). Já no caso de uma peça de Cp4 é feito o acionamento com nível 0 e é jogada na cesta 2, o segundo cluster.

Na figura 1 percebe-se que o os dois agrupamentos são linearmente separáveis, assim com algoritmo Perceptron podemos treinar uma rede que classifique corretamente as peças. Pela teoria a convergência é garantida para esse problema, assim podemos encontrar valores ajustados para os pesos e para polarização do neurônio que execute a discriminação das classes corretamente.

II. Parte 2

A segunda parte do trabalho o problema é semelhante mas o conjunto de peças que pertencem ao cesto 1 (nível 1) e cesto 2 (nível 0) se alteram seguindo a figura 2.

Saída	Conjuntos – Valor da Saída	Conjuntos – Valor da Saída
s1	Cp1,Cp2,Cp3 - 0	Cp4 – 1
s2	Cp1, Cp3,Cp4 – 0	Cp2 – 1
s3	Cp2,Cp3,Cp4 - 0	Cp1 – 1

Figura 2: Tabela de controle do robô para o item 2

Graficamente percebe-se que para cada saída, os conjuntos ainda são linearmente separáveis. Contudo, para cada caso a rede será novamente inicializada e treinada para obter os novos parâmetros do rede.

Em termos de desenvolvimento em código a função de aprendizagem será a mesma, pois segue as regras de implementação do Perceptron, mas o roteiro (*script*) de execução será alterado devido a mudança no parâmetro de treinamento, vetor saída/target.

III. Arquitetura

A arquitetura da rede neural do algoritmo Perceptron é similar ao de McCulloch and Pitts (1943), porém apresenta uma matriz de pesos e um vetor de polarização associado as entradas do neurônio.

As especificações do problema auxiliaram na definição da arquitetura final, no livro de Hagan et al. (1997) ele apresenta 3 passos para determinação da arquitetura:

1. Número de elementos do vetor (coluna) de entradas da rede = número padrões (características). Para o problema foram apresentada as medidas de duas características, logo esse número: $R = 2$

2. Número de neurônios na camada de saída = número de saídas do problema. Com o problema apresenta dois cestos para alocar as peças e a saída da rede funcionará como um decisor binário, logo o número de saídas corresponde a: $S = 1$

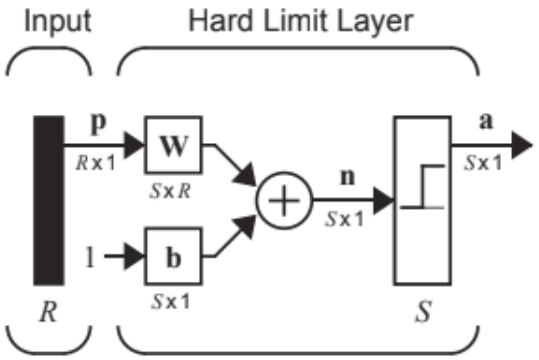
3. A escolha da função de ativação foi definida com a especificação no problema de como o robô é ativado com nível lógico 0 ou 1. Nesse caso a função é a mesma do algoritmo do

Perceptron, que no matlab corresponde a `hardlim`. E funciona da seguinte forma:

$$hardlim(n) = \begin{cases} 1 & \text{se } n \geq 0 \\ 0 & \text{caso contrário} \end{cases}$$

$$n = W' \cdot p + b$$

Assim a arquitetura da rede proposta equivale a apenas um neurônio com vetor de 2 elementos e uma saída binária (0 ou 1). A figura 3 representa graficamente o neurônio:



fonte: Adaptado de Hagan et al. (1997)

Figura 3: Arquitetura proposta, R = 2 e S =1

IV. Algoritmo Perceptron

A regra do perceptron funciona como a seguir.

- Há dois padrões de treinamento (características de entrada) p_i, temos um valor de saída desejada referente a classificação t_i.
- Uma saída a é calculada a partir dos valores de entrada. E com eles determinamos o erro e_i, que é a diferença do desejado com o obtido, $e_i = t - a$.
- Há um vetor de pesos inicializados e a polarização com valores aleatórios, nesse caso, foi definido no código uma distribuição normalizada. A cada saída os valores desses parâmetros são atualizados de acordo as formulas:
  1. Para os pesos: $w^{new} = w^{old} + e \cdot p$
  2. Para a polarização: $b^{new} = b^{old} + e$
- A rede é treinada com a execução finita dos passo anterior até anular o erro associado as entradas. As interações são finitas comprovadas pelo prova de Convergência do Perceptron que foi supracitado.

A seguir são apresentados os resultados com a classificação realizada pela rede treinada. Os códigos em MATLAB são apresentados no apêdince, sendo eles:

- `initNET.m` - Arquivo função-m responsável por inicializar a rede retornando a matriz de peso e a polarização. A geração dos valores é aleatória simétrica com o intervalo próximo de 0 e segue uma distribuição normal.
- `testNET.m` - Arquivo função responsável simular a rede dados os parâmetros de entrada, saída e pesos. Ela simplesmente executa a hardlim em todo o vetor de amostras de entrada e retorna um vetor associado de erros.
- `learnNET.m` - Arquivo função que executa o treinamento da rede usando laços de repetição para simular a rede e recalcular os pesos de modo que finalize quando o vetor de erros da amostra inicial seja formado completamente por 0's.
- `classificador1.m` - Arquivo script que executa de declara a amostra inicial do vetor de entradas, e realiza a invocação de cada procedimento na sequência correta. Também é responsável por plotar 3 gráficos: rede inicial, rede treinada, e de nível lógico associado as entradas

IV. Resultados

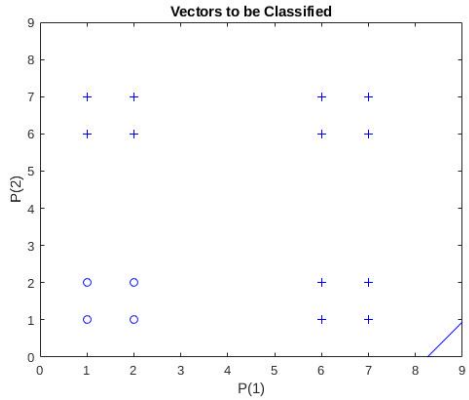
Neste capítulo apresentamos os resultados com a plotagem dos gráficos a amostra de entrdas, e o hiperplano traçado com os valores do peso e polarização em estado inicial e no estado após a conclusão do treinamento da rede.

I. Parte 1

Para cada inicialização da rede pelo método desenvolvido `init` temos um valor diferente, assim podemos ter o número de interações distintas. Mas para esse problema esse valor será sempre finito. Assim em uma das execuções gerou esse resultado descrito pelos gráficos abaixo.

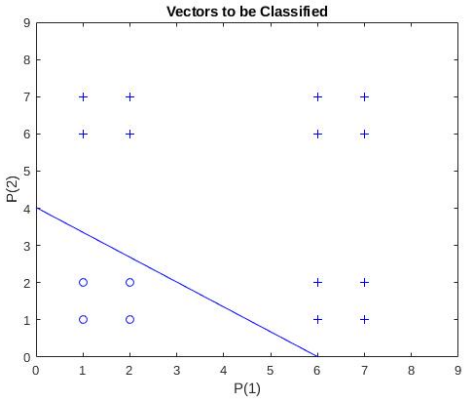
Para esse exemplo de treinamento da rede, obtivemos os seguinte valores:

- Inicialmente os valores de $W = \begin{bmatrix} -0.4336 \\ 0.3426 \end{bmatrix}$ e $b = 3.5784$.
- Foram realizadas exatamente 23 interações para encontrar os valores ajustados.
- Após o treinamento da rede os novos valores $W = \begin{bmatrix} 1.5664 \\ 2.3426 \end{bmatrix}$ e $b = -9.4216$



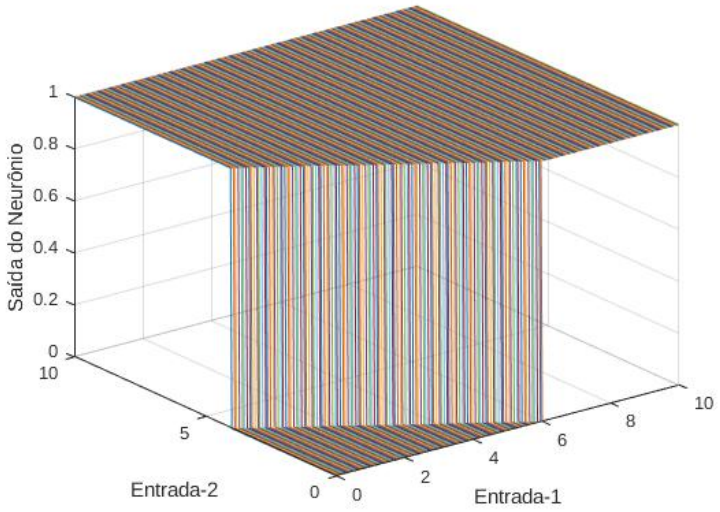
fonte: o Autor

(a) Pesos e Polarização inicializados



fonte: o Autor

(b) Rede Treinada



(c) Saída do neurônio pelas amostras de entradas

Figura 4: Os gráficos plotados para Cp1,Cp2,Cp3 - 1(+) e Cp4 - 0 (o)

fonte: o Autor

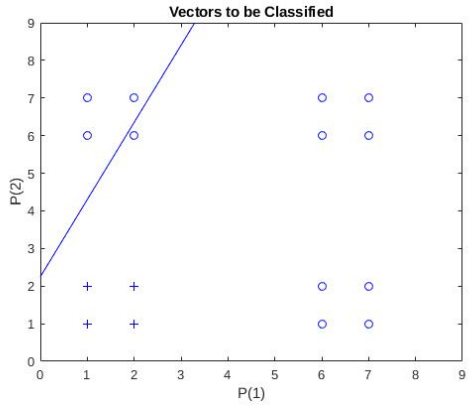
II. Parte 2

O roteiro de execução para a parte 2 é exatamente o mesmo com as mudanças apropriadas nos valores desejados para amostras de entradas.

II.1 Saída s1 - Cp1,Cp2,Cp3 - 0 e Cp4 - 1

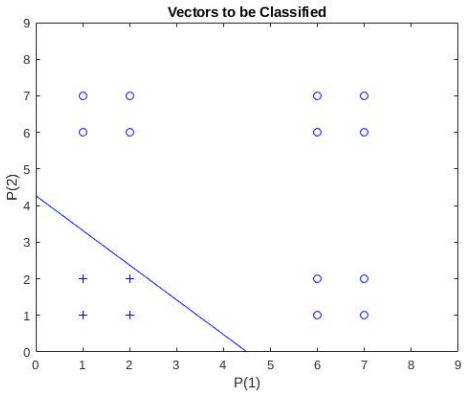
Com inicialização e treinamento da rede, obtivemos os seguinte valores:

- Inicialmente os valores de $W = \begin{bmatrix} 2.7694 \\ -1.3499 \end{bmatrix}$ e $b = 3.0349$.
- Foram realizadas exatamente 15 interações para encontrar os valores ajustados.



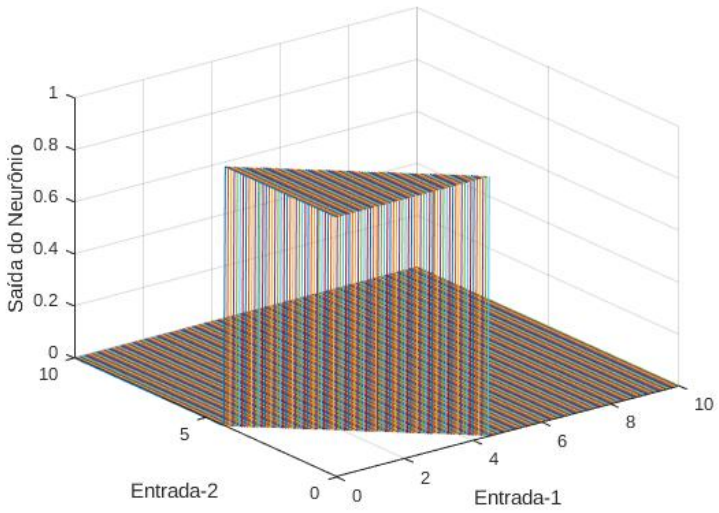
fonte: o Autor

(a) Pesos e Polarização inicializados



fonte: o Autor

(b) Rede Treinada



fonte: o Autor

(c) Saída do neurônio pelas amostras de entradas

Figura 5: s1 - Cp1,Cp2,Cp3 - 0 e Cp4 - 1

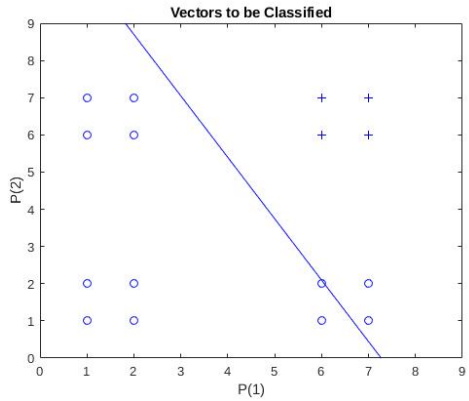
- Após o treinamento da rede os novos valores $W = \begin{bmatrix} -2.2306 \\ -2.3499 \end{bmatrix}$ e $b = 10.0349$

II.2 Saída s2 - Cp1,Cp3,Cp4 - 0 e Cp2 - 1

Para a segunda saída, com acionamento 1 para o conjunto de peças Cp2.

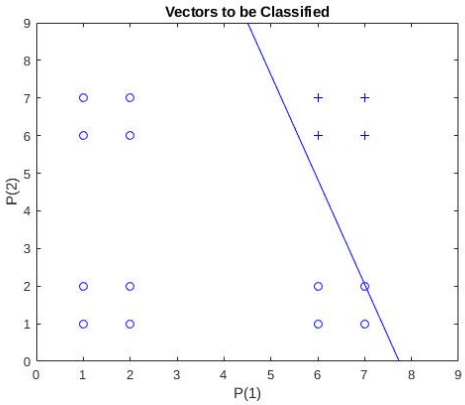
Com inicialização e treinamento da rede, obtivemos os seguinte valores:

- Inicialmente os valores de $W = \begin{bmatrix} -0.2050 \\ -0.1241 \end{bmatrix}$ e $b = 1.4897$.
- Foram realizadas exatamente 349 interações para encontrar os valores ajustados.
- Após o treinamento da rede os novos valores $W = \begin{bmatrix} 10.7950 \\ 3.8759 \end{bmatrix}$ e $b = -83.5103$



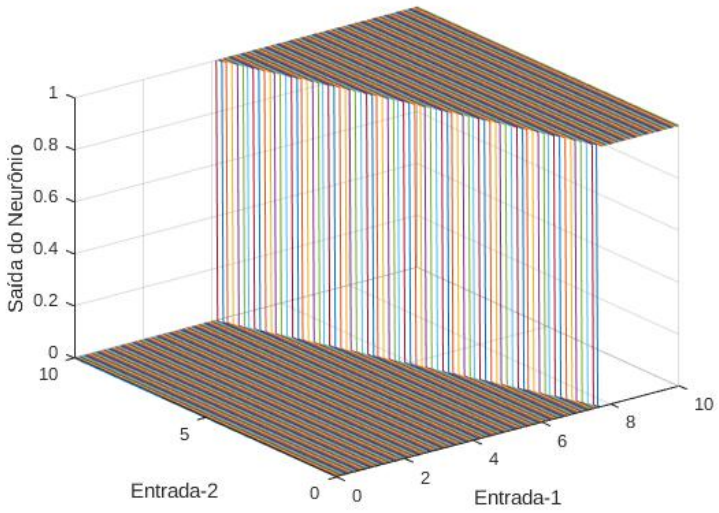
fonte: o Autor

(a) Pesos e Polarização inicializados



fonte: o Autor

(b) Rede Treinada



fonte: o Autor

(c) Saída do neurônio pelas amostras de entradas

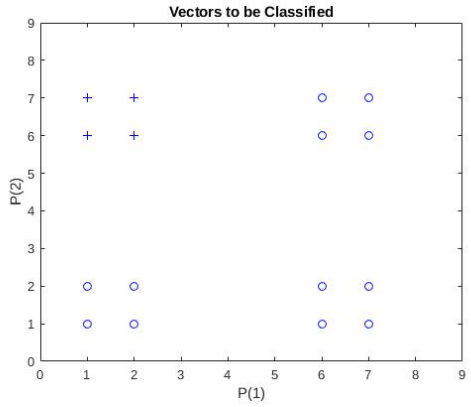
Figura 6: s2 - Cp1,Cp3,Cp4 - 0 e Cp2 - 1

II.3 Saída s3 - Cp1,Cp3,Cp4 - 0 e Cp2 - 1

Para a terceira saída, com acionamento 1 para o conjunto de peças Cp1.

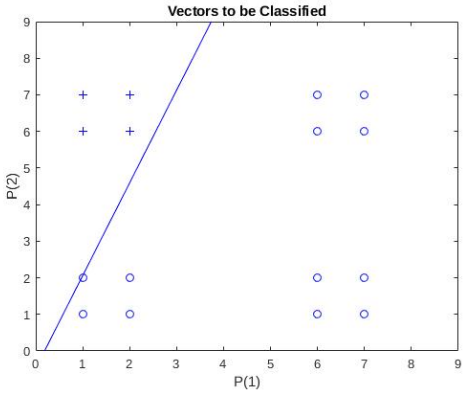
Com inicialização e treinamento da rede, obtivemos os seguinte valores:

- Inicialmente os valores de $W = \begin{bmatrix} 1.4090 \\ 1.4172 \end{bmatrix}$ e $b = 0.6715$.
- Foram realizadas exatamente 2 interações para encontrar os valores ajustados.
- Após o treinamento da rede os novos valores $W = \begin{bmatrix} -3.5910 \\ 1.4172 \end{bmatrix}$ e $b = 0.6715$



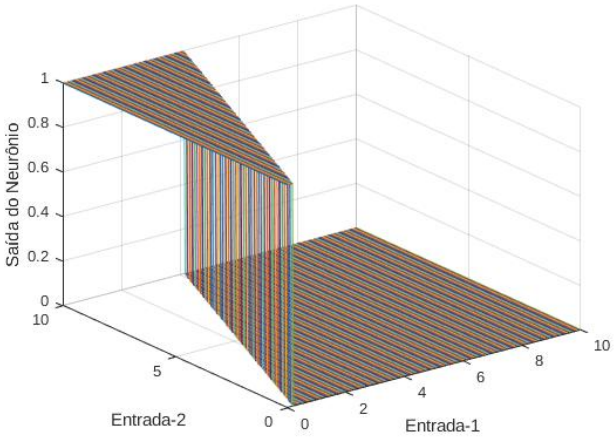
fonte: o Autor

(a) Pesos e Polarização inicializados



fonte: o Autor

(b) Rede Treinada



fonte: o Autor

(c) Saída do neurônio pelas amostras de entradas

Figura 7: s3 - Cp2,Cp3,Cp4 - 0 e Cp1 - 1

V. Conclusão

Neste trabalho realizamos o estudo e implementação de uma rede neural de 2 entradas e 1 camada. A rede é uma implementação do algoritmo Perceptron aplicado a situação-problema de classificação de um conjunto de peças. O desenvolvimento em MATLAB permitiu a representação do neurônio por um modelo matemático e dada uma intervalo de amostras de entrada, foi realizado o treinamento supervisionado da rede. Conclui-se que a arquitetura apresentada de uma camada é suficiente para classificar as peças corretamente. Para cada caso o número de interações realizadas variou de acordo com os valores iniciais dos pesos do neurônio. Cumprimos todos os objetivos que nos foi proposto, sendo este trabalho prático importante para o aprofundamento em sistemas conexionistas, pois permitiu aplicar a teoria para um problema prático e fixar uma das primeiras técnicas de aprendizagem supervisionado e reconhecimento de padrões.

Referências

D. R. Guha and S. K. Patra. Cochannel interference minimization using wilcoxon multilayer perceptron neural network. In *2010 International Conference on Recent Trends in Information, Telecommunication and Computing*, pages 145–149, March 2010. doi: 10.1109/ITC.2010.50.

M. T. Hagan, H. B. Demuth, M. Hudson, et al. Neural network design. 1997.

M. L. Johnson. Perceptron-how this neural network model lets you evaluate boolean functions. *IEEE Potentials*, 12(3):17–18, Oct 1993. ISSN 0278-6648. doi: 10.1109/45.282290.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

M. Minsky and S. Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

VI. Apêndice

Neste Apêndice encontram-se os códigos fonte dos scprits e funções implementados em MATLAB utilizadas neste trabalho para realizar a especificação e treinamento da rede, desenvolvido para realizar a classificação de peças de acordo com duas características já mensuradas.

I. initNET

```
function [W,b] = initNET(P,T)
%INITNET Retorna a matriz de pesos e a polarização
%   Gera valores iniciais para os pesos e as polarização de um único
%   neurônio, sendo feito por um número aleatório simétrico próximo de
%   zero. Sendo estes números normalmente distribuídos.
%   P = RxQ matriz dos vetores de entrada (P1..PQ), Pi = [p1..pR].  #1
%   T = SxQ vetor de saídas (t1..tQ), S = 1 para uma camada  #2

% R - entradas da rede neural
[R , Qp] = size(P);
[S , Qt] = size(T);
if(Qp == Qt)
    W = randn(R,1); % Matriz SxR , S = 1, somente um neuronio
    b = randn(); % Matriz Sx1, S = 1, idem
else
    error('Deve haver correspondência de Q colunas de entradas e saídas(targets).');
end
```

II. testNET

```
function [E] = testNET(P,T,W,b)
%LEARNNET Regra unificada de aprendizagem do perceptron
%   Calcula as variações nos pesos e polarizações de um conjunto de
%   entradas P e o erro.
%    P = RxQ matriz dos vetores de entrada (P1..PQ), Pi = [p1..pR].  #1
%    T = SxQ vetor de saídas (t1..tQ), S = 1 para uma camada  #2
%    A = RxQ matriz dos vetores de saída com a aplicação da hardlim(W'A+b).
%    E = SxQ matriz dos vetores de erro.


n = W'*P +b;
a = hardlim(n);
E = T - a;
end
```

III. learnNET

```
function [Wn, bn] = learnNET(P,T,W,b)
%LEARNNET Regra unificada de aprendizagem do perceptron
%   Calcula as variações nos pesos e polarizações de um conjunto de
%   entradas P e o erro.
%    P = RxQ matriz dos vetores de entrada (P1..PQ), Pi = [p1..pR].  #1
%    T = SxQ vetor de saídas (t1..tQ), S = 1 para uma camada  #2
%    A = RxQ matriz dos vetores de saída com a aplicação da hardlim(W'A+b).
%    E = SxQ matriz dos vetores de erro.

cont = 0;
Wn = W;
bn = b;
E = testNET(P,T,Wn,bn);
disp(E);
while max(E) == 1 || min(E) == -1
        for i = 1:length(P) % 1 até Q
           if(E(i) ~= 0)
           disp(E(i))
               Wn = Wn + E(i)*P(:,i)  ;    % acessa o valor do vetor Pi
               bn = bn + E(i);
               break;
           end
        end
    E = testNET(P,T,Wn,bn);
    disp(E);
```

```
    cont = cont +1;
end
disp(cont);
end
```

IV. Scripts de classificação

IV.1 Parte 1 - Classificador

```
close all,clear all,clc, format compact;

% P_i = [ caracteristica_1; caracteristica_2]
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |
P = [ 1 1 2 2 6 6 7 7 6 6 7 7 1 1 2 2;
      7 6 7 6 7 6 7 6 1 2 1 2 1 2 1 2 ];
T = [ 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 ];
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |

% Inicializando os valores de pesos e polarização
[W, b] = initNET(P,T);

% Tentativa inicial de classificação
f1 = figure('Name','Pesos e Polarização incializados');
plotpv(P,T, [0 9 0 9]) ;
plotpc(W', b);


[Wn,bn] = learnNET(P,T,W,b);
disp(Wn);
disp(bn);
% Plotagem
f2 = figure('Name','Rede Treinada');
plotpv(P,T, [0 9 0 9]) ;
plotpc(Wn', bn);

f3 = figure('Name','saída do neurônio sobre o intervalo de entradas');
[p1,p2] = meshgrid(0:0.05:10);
z = feval('hardlim',[p1(:) p2(:)]*Wn+bn);
z = reshape(z,length(p1),length(p2));
plot3(p1,p2,z);
grid on
xlabel('Entrada-1');
ylabel('Entrada-2');
zlabel('Saída do Neurônio');
```

IV.2 Parte 2 - Classificador s1

```
close all,clear all,clc, format compact;

% P_i = [ caracteristica_1; caracteristica_2]
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |
P = [ 1 1 2 2 6 6 7 7 6 6 7 7 1 1 2 2;
      7 6 7 6 7 6 7 6 1 2 1 2 1 2 1 2 ];
T = [ 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 ];
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |

% Inicializando os valores de pesos e polarização
[W, b] = initNET(P,T);

% Tentativa inicial de classificação
f1 = figure('Name','Pesos e Polarização incializados');
plotpv(P,T, [0 9 0 9]) ;
plotpc(W', b);


[Wn,bn] = learnNET(P,T,W,b);
disp(Wn);
disp(bn);
% Plotagem
f2 = figure('Name','Rede Treinada');
plotpv(P,T, [0 9 0 9]) ;
plotpc(Wn', bn);

f3 = figure('Name','saída do neurônio sobre o intervalo de entradas');
[p1,p2] = meshgrid(0:0.05:10);
z = feval('hardlim',[p1(:) p2(:)]*Wn+bn);
z = reshape(z,length(p1),length(p2));
plot3(p1,p2,z);
grid on
xlabel('Entrada-1');
ylabel('Entrada-2');
zlabel('Saída do Neurônio');
```

IV.3 Parte 2 - Classificador s2

```
close all,clear all,clc, format compact;

% P_i = [ caracteristica_1; caracteristica_2]
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |
P = [ 1 1 2 2 6 6 7 7 6 6 7 7 1 1 2 2;
```

```
      7 6 7 6 7 6 7 6 1 2 1 2 1 2 1 2 ];
T = [ 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 ];
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |

% Inicializando os valores de pesos e polarização
[W, b] = initNET(P,T);

% Tentativa inicial de classificação
f1 = figure('Name','Pesos e Polarização incializados');
plotpv(P,T, [0 9 0 9]) ;
plotpc(W', b);


[Wn,bn] = learnNET(P,T,W,b);
disp(Wn);
disp(bn);
% Plotagem
f2 = figure('Name','Rede Treinada');
plotpv(P,T, [0 9 0 9]) ;
plotpc(Wn', bn);

f3 = figure('Name','saída do neurônio sobre o intervalo de entradas');
[p1,p2] = meshgrid(0:0.05:10);
z = feval('hardlim',[p1(:) p2(:)]*Wn+bn);
z = reshape(z,length(p1),length(p2));
plot3(p1,p2,z);
grid on
xlabel('Entrada-1');
ylabel('Entrada-2');
zlabel('Saída do Neurônio');
```

IV.4 Parte 2 - Classificador s3

```
close all,clear all,clc, format compact;

% P_i = [ caracteristica_1; caracteristica_2]
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |
P = [ 1 1 2 2 6 6 7 7 6 6 7 7 1 1 2 2;
      7 6 7 6 7 6 7 6 1 2 1 2 1 2 1 2 ];
T = [ 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ];
%    |  Cp1  |  Cp2  |  Cp3  |  Cp4  |

% Inicializando os valores de pesos e polarização
[W, b] = initNET(P,T);
```

```
12
13 % Tentativa inicial de classificação
14 f1 = figure('Name','Pesos e Polarização incializados');
15 plotpv(P,T, [0 9 0 9]) ;
16 plotpc(W', b);
17
18
19 [Wn,bn] = learnNET(P,T,W,b);
20 disp(Wn);
21 disp(bn);
22 % Plotagem
23 f2 = figure('Name','Rede Treinada');
24 plotpv(P,T, [0 9 0 9]) ;
25 plotpc(Wn', bn);
26
27 f3 = figure('Name','saída do neurônio sobre o intervalo de entradas');
28 [p1,p2] = meshgrid(0:0.05:10);
29 z = feval('hardlim',[p1(:) p2(:)]*Wn+bn);
30 z = reshape(z,length(p1),length(p2));
31 plot3(p1,p2,z);
32 grid on
33 xlabel('Entrada-1');
34 ylabel('Entrada-2');
35 zlabel('Saída do Neurônio');
```