

# Trabalho 4: Backpropagation

PEDRO VICTOR DOS SANTOS MATIAS, 21601225

pvs@icomp.ufam.edu.br

## I. INTRODUÇÃO

O presente trabalho é sobre a redes multicamadas com algoritmo de *backpropagation*. O problema abordado neste trabalho apresentará a flexibilidade do perceptron multicamadas para implementar uma função sinusoidal. O algoritmo para atualizar a rede, *backpropagation*, é uma generalização do LMS com a complexidade de calcular as derivadas parciais devido a camada oculta da rede.

Este documento está organizado em 6 seções. Na segunda, será realizado uma revisão bibliográfica sobre trabalhos envolvendo o perceptron multicamadas (MLP) e o algoritmo de *backpropagation*. No capítulo 3 é descrita a arquitetura geral da rede e as perguntas a serem respondidas. Os resultados, como gráficos e parâmetros da rede obtidos são apresentados na seção 4. Na seção de conclusão é realizado uma as considerações finais entre o problema e os resultados obtidos. E por último um apêndice com a listagem dos códigos em MATLAB.

## II. REVISÃO BIBLIOGRÁFICA

Como mencionamos na seção anterior, o trabalho está diretamente relacionado a aproximação de uma função a partir de um perceptron de multicamadas com o algoritmo de *backpropagation* para realizar a aprendizagem. Sendo assim, apresentamos alguns trabalhos que envolvem a arquitetura de rede ou a regra de aprendizagem usadas neste experimento.

Uma das principais áreas de aplicação do MLP é treinamento de redes não-lineares. As principais desvantagens dos algoritmos de treinamento convencionais são a estagnação ótima local e a lenta velocidade de convergência. Isso torna o algoritmo de otimização estocástica uma alternativa confiável para aliviar esses inconvenientes. O trabalho de Aljarah et al. (2018) propõe um novo algoritmo de treinamento baseado no algoritmo de otimização de baleias (WOA) proposto recentemente. Ficou provado que este algoritmo é capaz de resolver uma ampla gama de problemas de otimização e supera os algoritmos atuais. Isso motivou nossas tentativas de avaliar seu desempenho no treinamento de redes neurais feedforward. Pela primeira vez na literatura, um conjunto de 20 conjuntos de dados com diferentes níveis de dificuldade foram escolhidos para testar a rede baseada na regra WOA. Os resultados são verificados por meio de comparações com o algoritmo de *backpropagation* e seis técnicas evolutivas. Os resultados qualitativos e quantitativos provaram que o algoritmo proposto é capaz de superar os algoritmos atuais na maioria dos conjuntos de dados em termos de otimização local e velocidade de convergência.

No artigo de Durairaj and Revathi (2015) é apresentado como problema no diagnóstico da presença de doença cardíaca por ser um processo realmente tedioso, pois requer conhecimento profundo. É citado que a predição de doença cardíaca se baseia na maneira tradicional de examinar o relatório médico, como ECG (O Eletrocardiograma), Ressonância Magnética

(Ressonância Magnética), Pressão Arterial, Testes de estresse por um médico. O trabalho teve como objetivo prever a existência de doenças cardíacas usando o backpropagation em um MLP (Perceptron Multicamada). Os resultados experimentais mostraram que o MLP com algoritmo de treinamento adequado pode ser uma ferramenta eficaz para prever a doença cardíaca com melhor precisão.

### III. METODOLOGIA

Considerando a função abaixo:

$$f(x) = 1 + \cos\left(\frac{6\pi}{4}p\right), \text{ para } -2 < p < 2$$

Pede-se para aproximar essa função através de uma rede neural com uma arquitetura 1-S-1, utilizando uma função de transferência log-sigmoide nos neurônios da primeira camada e uma função linear no neurônio da segunda camada sem utilização das funções de alto nível do Matlab. Como resultados de saída são pedidos:

- i Gerar 50 pontos igualmente espaçados entre -2 e 2;
- ii Utilizar os pontos ímpares para treinamento e os pares para teste;
- iii Realizar o treinamento e determinar a menor quantidade de neurônios na camada intermediária,  $n$ , para a qual a função é aproximada com razoável precisão;
- iv Para a rede com a quantidade de neurônios na camada intermediária determinada no item 3, calcular o valor do erro médio quadrático no conjunto de teste;
- v Traçar o gráfico da função e o gráfico da rede com a quantidade de neurônios na camada intermediária determinada no item 3, superpostos em um mesmo gráfico;

Para resolver essas questões será usado os conceitos aprendidos sobre a arquitetura da rede MLP e o algoritmo de aprendizagem backpropagation. No próximo capítulo as equações e resultados serão apresentados.

## IV. RESULTADOS

A rede proposta é a seguinte

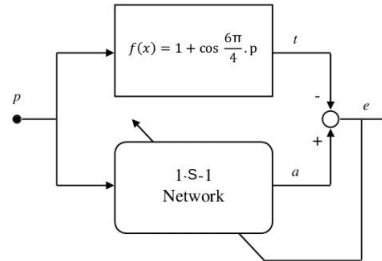


Figura 1: Arquitetura

Primeiramente, os 50 pontos gerados para o conjunto de de treinamento e teste são igualmente espaçados.

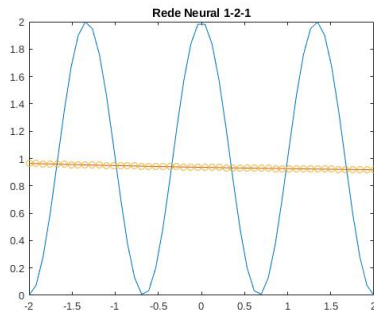
-2,0000	-1,9184	-1,8367	-1,7551	-1,6735	-1,5918	-1,5102	-1,4286	-1,3469	-1,2653
-1,1837	-1,1020	-1,0204	-0,9388	-0,8571	-0,7755	-0,6939	-0,6122	-0,5306	-0,4490
-0,3673	-0,2857	-0,2041	-0,1224	-0,0408	0,0408	0,1224	0,2041	0,2857	0,3673
0,4490	0,5306	0,6122	0,6939	0,7755	0,8571	0,9388	1,0204	1,1020	1,1837
1,2653	1,3469	1,4286	1,5102	1,5918	1,6735	1,7551	1,8367	1,9184	2,0000

Tabela 1: 50 pontos entre -2 e 2

Foi separados dois conjuntos com esses pontos, para o de treinamento aqueles com valores ímpares e os pares para o de teste.

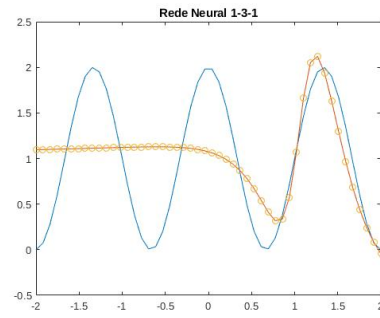
### I. Treinamento

Para o treinamento, foi realizado vários testes e definiu o valor de épocas de 12000. Isto é foram realizadas 12000 iterações para cada arquitetura de rede desde 1-2-1 até 1-6-1 que obtivemos resultados mais satisfatório com erro de médio quadrático  $e = 1.7574E - 04$



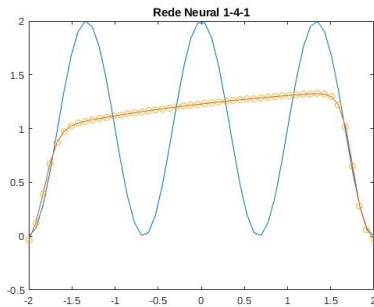
fonte: o Autor

(a) Arquitetura 1-2-1



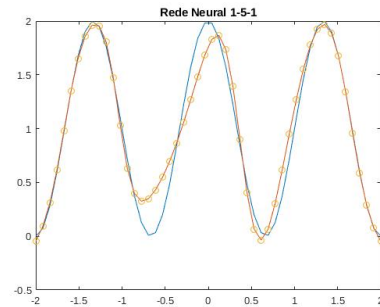
fonte: o Autor

(b) Arquitetura 1-3-1



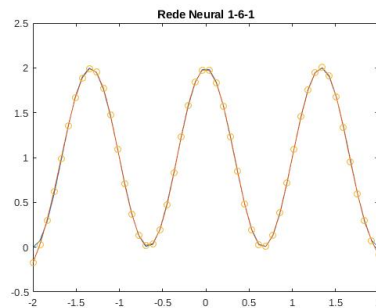
fonte: o Autor

(c) Arquitetura 1-4-1



fonte: o Autor

(d) Arquitetura 1-5-1



fonte: o Autor

(e) Arquitetura 1-6-1

Figura 2: Plotagem do conjunto de Teste para Redes 1-S-1 , com S = 2 a 6

Após as 12000 iterações na rede 1-6-1 obtivemos resultados satisfatórios após o treinamento, como mostrado no teste. Ou seja, a rede conseguiu aprender o padrão da função e atualizou os pesos como o resultado da aproximação, esses pesos e as polarizações obtidos foram:

$$W^1 = \begin{bmatrix} -5,235437 \\ -4,492094 \\ 6,121017 \\ -5,926790 \\ 6,082437 \\ -5,333465 \end{bmatrix} \quad b^1 = \begin{bmatrix} 1,9675 \\ 4,4041 \\ 1,9861 \\ -6,0384 \\ -9,7541 \\ -8,7284 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 4,902607 & -5,457951 & 3,770000 & 3,939589 & -3,878252 & -3,780106 \end{bmatrix} \quad b^2 = -0,2425$$

O algoritmo desenvolvido em matlab realiza os seguintes passos:

- 1 inicializar a matriz de pesos e das polarizações de cada camada com valores randômicos normalmente distribuídos em torno do zero;
- 2 realizar a propagação direta com cada amostra de treinamento
- 3 calcular o erro
- 4 calcular a sensibilidade
- 5 atualizar os pesos e as polarizações de cada camada.
- 6 repetir o passo 2 até atingir o número de épocas pré-definido.

## V. CONCLUSÃO

Neste trabalho realizamos o estudo e implementação de uma rede neural Perceptron Multi-camada e usamos como regra de aprendizagem o algoritmo de backpropagation que é semelhante ao LMS por ser estocástico, mas um pouco mais complexo devido os cálculos que são propagados pela camada oculta. A condição de parada definida no desenvolvimento foi o número de épocas, 12,000, e para arquitetura 1-6-1 obtivemos sucessos com o conjunto de testes. Um procedimento que poderia ser melhorado seria o processo de seleção do elementos do conjunto de treinamento que eram sequenciais, e poderiam ser distribuídos de forma aleatória. No desenvolvimento houve vezes que a própria arquitetura 1-6-1 não convergia isso devido o valor inicializado, uma solução seria usar como condição de parada o erro quadrático médio e não número de épocas. A taxa de aprendizagem seguiu informação do próprio livro-texto, Hagan et al. (1997), de 0,1 não tendo sido testado nenhum outro valor. Sendo assim, o experimento foi executado com sucesso, contudo algumas melhorias poderiam ter sido realizadas para melhor aproveitamento das condições do problema.

## REFERÊNCIAS

- I. Aljarah, H. Faris, and S. Mirjalili. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, 22(1):1–15, 2018.
- M. Durairaj and V. Revathi. Prediction of heart disease using back propagation mlp algorithm. *International Journal Of Scientific & Technology Research*, 4(08), 2015.
- M. T. Hagan, H. B. Demuth, and M. Beale. Neural network design. 1997.

## VI. APÊNDICE

Neste Apêndice encontram-se os códigos fonte do script implementado em MATLAB utilizado neste trabalho para realizar a especificação e treinamento da rede, desenvolvido para realizar a função de filtro preditivo.

### I. Script

```

1  % Trabalho 4.
2  % Pedro V D S Matias (pvs@icompu.ufam.edu.br), 30-05-2019 12:29
3  %-----
4  close all,clear all,clc;
5
6
7  % Rede Neural arquitetura 1-S-1
8  % 1) Gerar 50 pontos igualmente espaçados entre -2 e 2;
9  p = linspace(-2,2,50)';
10
11 % 2) Utilizar os pontos ímpares para treinamento e os pares para teste;
12 p_tr = []; p_tt = []; % vetor pontos de treinamento e teste
13 for i=1:length(p)
14     if mod(p(i),2) ~= 0 % ímpares
15         p_tr(end+1,1) = p(i);
16     else
17         p_tt(end+1,1) = p(i);
18     end
19 end
20 % 3) Realizar o treinamento e determinar a menor quantidade de neurônios na
21 % camada intermediária, n, para a qual a função é aproximada com razoável precisão;
22 %f = 1 + cos(p*6*pi/4);
23 % W[S,R], Matriz de pesos e b[S,1] - vetor de polarização, sendo R = 1
24 R = 1; S1 = 6; S2 = 1;
25 W1 = randn(S1,R);    b1 = randn(S1,1);
26 W2 = randn(S2,S1);   b2 = randn(S2,1);
27 save('W1_inicial.txt','W1','-ascii'); % Para salvar os dados
28 save('W2_inicial.txt','W2','-ascii');
29 save('b1_inicial.txt','b1','-ascii');
30 save('b2_inicial.txt','b2','-ascii');
31 epoca_tr = 12000;
32 E = zeros(1,epoca_tr);
33
34
35 for k = 1:epoca_tr
36     disp(k);

```

```

37     %x = randsample(p_tr,length(p_tr));
38     x = p_tr;
39     for i = 1:length(p_tr)
40
41         % Forward Propagation
42         a0 = x(i);
43         t = 1 + cos(a0*6*pi/4);
44         % Saída da primeira camada oculta
45         n1 = W1*a0 + b1;
46         a1 = logsig(n1);
47
48         % Saída da segunda camada
49         n2 = W2*a1 + b2;
50         a2 = purelin(n2);
51
52         % O erro será
53         e = (t - a2);
54
55         % Backward Propagation
56         % derivadas das funções de transferência e sensibilidade
57         F2 = diag(dpurelin(n2,a2));
58         s2 = -2 * F2 * e;
59         F1 = diag(dlogsig(n1,a1));
60         s1 = F1 * W2' * s2;
61
62         % Para simplificar, usaremos uma taxa de aprendizado 0,1
63         alpha = 0.1;
64         W2 = W2 - alpha .* s2 * a1';
65         b2 = b2 -alpha .* s2;
66
67         W1 = W1 - alpha .* s1 * a0';
68         b1 = b1 - alpha .* s1;
69
70     end
71 end
72 % Teste da rede
73 g = [];
74 for i = 1: length(p)
75     % Saída da primeira camada oculta
76     a0 = p(i);
77     n1 = W1*a0 + b1;
78     a1 = logsig(n1);
79     % Saída da segunda camada

```

```
80     n2 = W2*a1 + b2;
81     a2 = purelin(n2);
82     g(end+1,1) = a2;
83 end
84
85 %figure;
86 y = 1 + cos(p*6*pi/4);
87 plot(p, y)
88
89 title('Rede Neural 1-6-1')
90
91 hold on
92
93 plot(p,g)
94 scatter(p,g)
95 hold off
```