Otimização de Funções com Métodos de Pesquisa

Protocolo de Trabalho Prático 3

INTELIGÊNCIA ARTIFICIAL

4 de dezembro de 2022

Criado por: Francisco Conceição al73819

utad

Inteligência Artificial
Licenciatura em Engenharia Informática
ECT - 2022/2023

Otimização de Funções com Métodos de Pesquisa

Protocolo de Trabalho Prático 3

1.Introdução

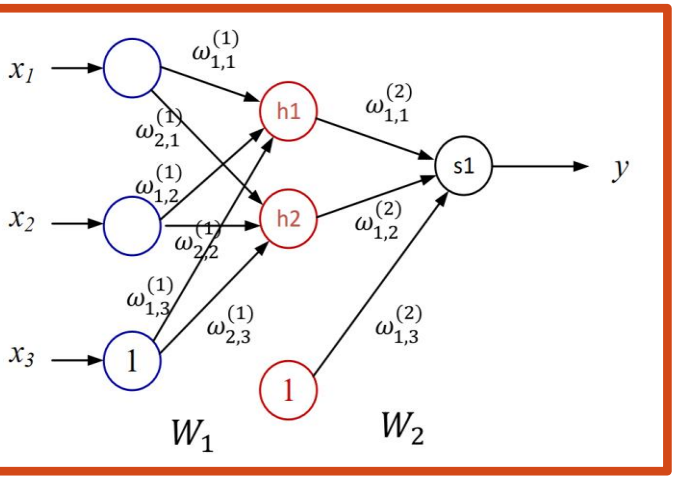
No âmbito da Unidade Curricular de Inteligência Artificial, do curso de Engenharia Informática foi proposto pelos regentes da mesma a realização de um protocolo sobre as Redes Neuronais Artificiais, conteúdo aprendido nas aulas teóricas desta mesma UC. Este relatório é dividido em duas partes, a primeira parte aborda a Rede Neural de formação 3x3x1 e a segunda parte a Rede Neural 3x5x3 com classificação das classes.

2.Objetivos do Trabalho

O objetivo principal deste trabalho é o desenvolvimento da aquisição de conhecimentos e de competências relativas às redes neuronais artificiais, às suas regras de aprendizagem e às Redes Neuronais Multicamada.

3.Classificação de dados não linearmente separáveis

Na primeira parte do relatório é nos pedido um programa que treina a rede neuronal de topologia (3x3x1) apresentada na figura a seguir, de alimentação direta de forma a classificar a função lógica ou-exclusivo, função esta apresentada na tabela a seguir considerando duas entradas e uma entrada adicional de bias(+1).



1-Rede Neuronal de topologia 3x3x1

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

2-Tabela de verdade do Ou-Exclusivo

Para a construção consideram a taxa de aprendizagem α=0.8, os pesos inicializados aleatoriamente no intervalo [-1, 1] e 2000 épocas de treino, que ao longo deste relatório irá sofrer alterações.

Com os seguintes dados apresentados, com a ajuda do vídeo mencionado na bibliografia e da ferramenta Matlab, foi construído o programa que se encontra no ficheiro zipado com o nome "mainNaoLinearmenteSeparaveis.m".

3.1. Resultados

Na primeira simulação os pesos inicializados aleatoriamente obtiveram estes valores:

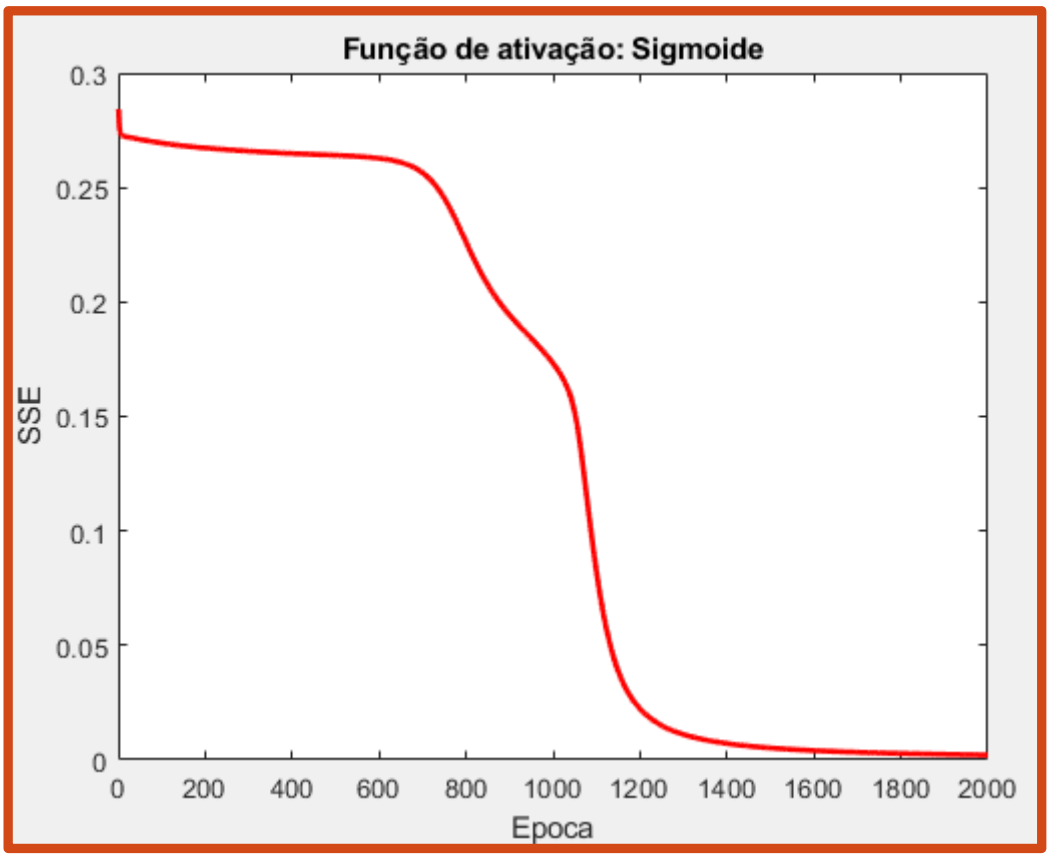
$$W1 = \begin{matrix} 0.4079 & 0.3753 & -0.2383 \\ 0.8646 & 0.1367 & 0.2692 \end{matrix}$$

$$W2 = -0.2735 \quad -0.1848 \quad -0.2626$$

E no final do programa obteve-se a função de ativação Sigmoide e os valores dos pesos:

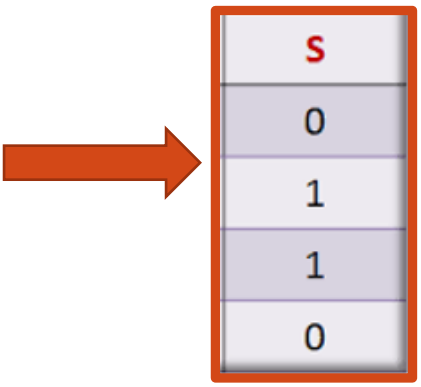
$$W1 = \begin{matrix} 5.0326 & -4.8168 & 2.3744 \\ 5.8417 & -5.9909 & -3.2414 \end{matrix}$$

$$W2 = -7.3844 \quad -7.5934 \quad -3.24.14$$



Podemos concluir que esta simulação foi bem-sucedida pois o vetor de soma dos erros quadráticos foi diminuído ao longo das épocas, chegando a ser quase nulo. E como apresentado na figura a seguir ao aproximar o resultado dos Y é igual ao resultado da função ou-exclusivo.

$$Y = \begin{matrix} 0.0466 \\ 0.9460 \\ 0.9586 \\ 0.0403 \end{matrix}$$



S
0
1
1
0

Na segunda simulação, foi feita com um número elevado de épocas (10000), o valor expectável desta simulação era que com o aumento do número de épocas o erro diminui-se e o resultado aproxima-se mais com a matriz target (resultado da função lógica XOR). Os pesos inicializados aleatoriamente tiveram os valores:

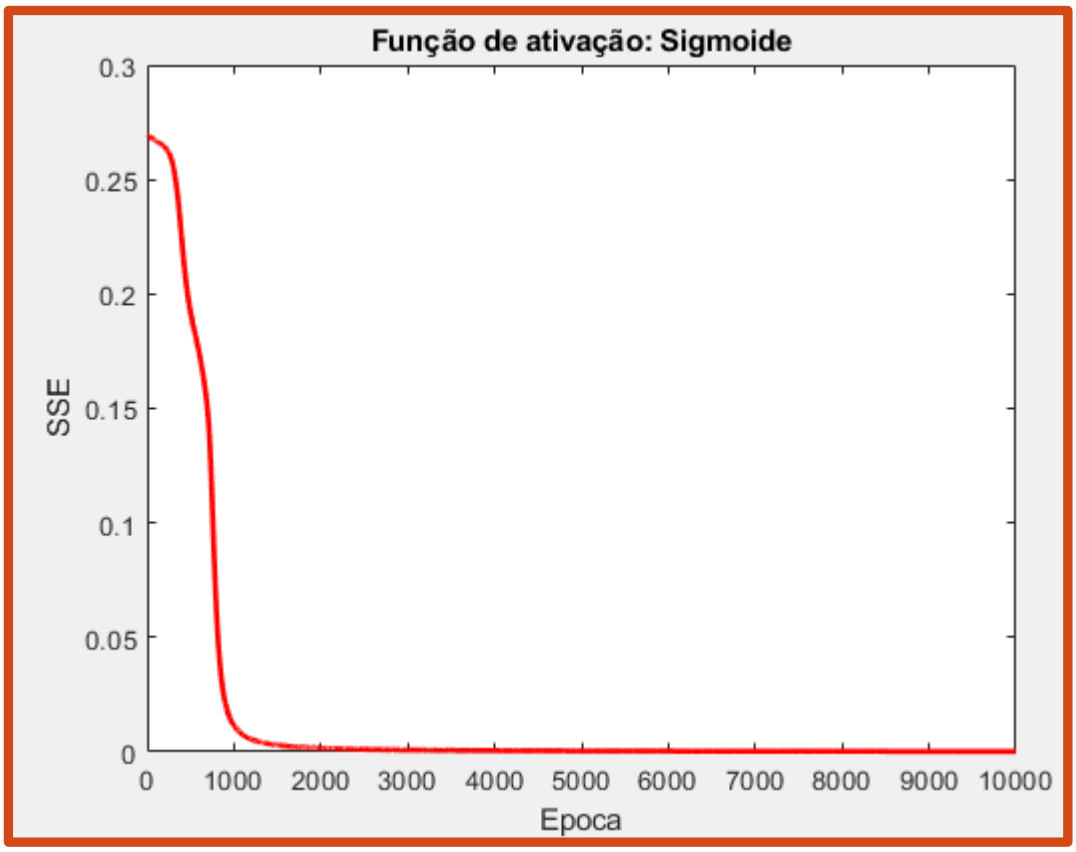
$$W1 = \begin{matrix} -0.0632 & 0.8211 & -0.3228 \\ 0.0068 & -0.5871 & 0.1483 \end{matrix}$$

$$W2 = -0.0261 \quad -0.4756 \quad -0.1592$$

Com os pesos e os dados inicializados, o FeedForward, o BackPropagation, atualização dos pesos e o teste da rede, os resultados foram os seguintes:

$$W1 = \begin{matrix} -4.9700 & -4.9564 & 7.3946 \\ -6.6483 & -6.5802 & 2.7322 \end{matrix}$$

$$W2 = 10.2857 \quad -10.4199 \quad -4.8948$$



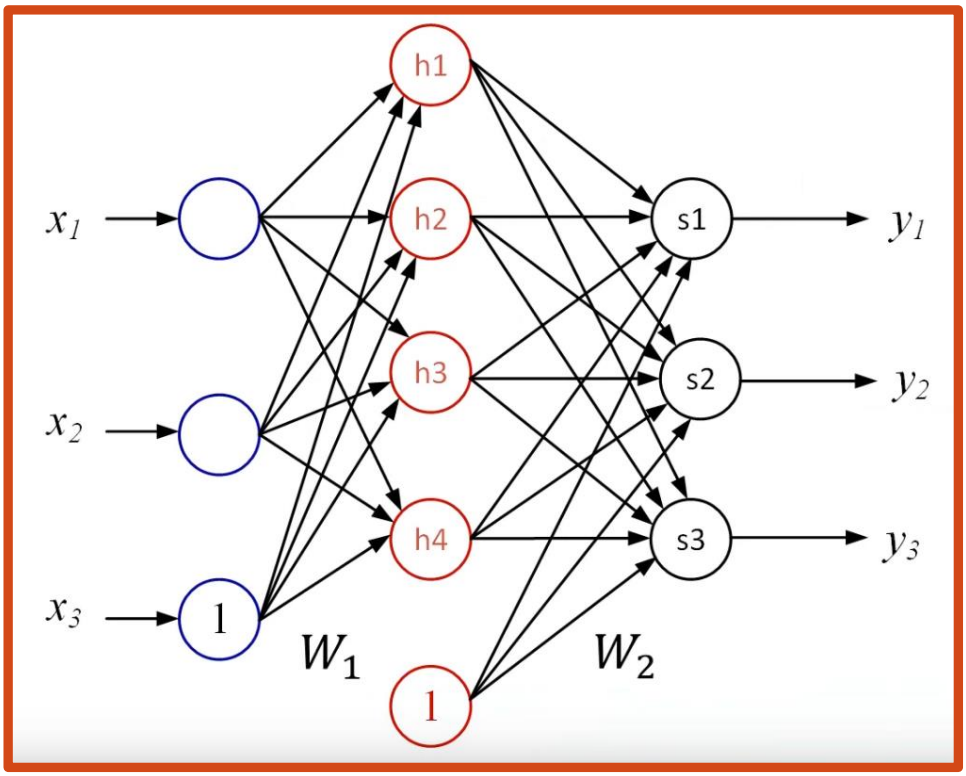
Com os resultados obtidos, existe uma diminuição gigante do erro comparado com a função resultada das 2000 épocas como já esperado, e a mudança expectável dos pesos. O Y ficou mais aproximado do Target comparando-o com a simulação anterior, concluindo assim que com um aumento do numero de épocas o Y aproxima-se mais do resultado

$$Y = \begin{matrix} 0.0121 \\ 0.9872 \\ 0.9873 \\ 0.0157 \end{matrix}$$

S
0
1
1
0

5.Classificação multi-classe

Na segunda parte do relatório, foi pedido para treinar uma RN com topologia 3x5x4, e utilizá-la para classificar padrões com 3 classes com base num conjunto de dados de treino.



Considerando que os valores desejados (target) para cada uma das três classes é representado usando valores binário:

- **Classe A – [1 0 0];**
- **Classe B – [0 1 0];**
- **Classe C – [0 0 1]:**

Para o treinamento da rede consideramos que das 120 amostras do conjunto de treino dado, origina 40 amostras para cada classe. Utilizando como pedido a função de ativação sigmoide, uma taxa de aprendizagem $\alpha = 0.9$, e os pesos que são iniciados aleatoriamente no intervalo [-1, 1] e 10000 épocas de treino.

Foi usado para testar a rede o conjunto de dados seguinte:

$$X_{Teste} = \begin{matrix} 0.5 & 1.0 \\ 0.4 & 1.2 \\ -0.5 & 0.5 \\ 1.0 & 1.0 \\ 1.0 & 1.5 \\ 0.55 & 0.55 \\ 1.0 & 0.0 \\ 0.55 & 0.55 \\ 1.0 & 0.0 \\ 0.5 & 0.0 \\ 1.5 & 0.0 \\ 0 & 0.5 \end{matrix}$$

Com os seguintes dados apresentados, com a ajuda do vídeo mencionado na bibliografia e da ferramenta Matlab, foi construído o programa que se encontra no ficheiro zipado com o nome "mainDataMulti.m".

Para o teste da rede foi implementado o seguinte código:

```
N = 10;                                    1

yplot=zeros(N,3);
%Acrescentar Bios à matriz XTeste          2
XTeste_Bios = ones(N,1);
X_teste = [X_teste, XTeste_Bios];

for k = 1:N
   x_teste = X_teste(k,:)';
   g1=W1*x_teste;
   %Sigmoide
   y1 = sig(g1);
   %y1 adicionada com entrada de bias
   y1_b = [y1
           1];                             3
   %Soma e saída
   g2 = W2*y1_b;
   y2= sig(g2);
   %Para o grafico
   y_plot(k,:) = y2;

end
```

```
%Limite para a classificação
limite = 0.9;                              4
y_bin = y_plot>limite;

for k = 1:N
    display(y_bin(k, :));
    display(X_teste(k,1));
    display(X_teste(k,2));                 5
    if y_bin(k, :) == [1 0 0]
        plot(X_teste(k,1), X_teste(k,2), 'bd');
    elseif y_bin(k, :) == [0 1 0]
        plot(X_teste(k,1), X_teste(k,2), 'rd');
    elseif y_bin(k, :) == [0 0 1]
        plot(X_teste(k,1), X_teste(k,2), 'kd');
    else
        plot(X_teste(k,1), X_teste(k,2), 'md');
    end
end
hold off;
```

No ponto **1**, definimos o número de amostras da matriz teste, neste caso é 10, mas podiam ser mais. Depois definindo a matriz *yplot* onde vão ser guardados todos os Y resultado da rede a cada linha da matriz teste, construiu-se a matriz BIOS juntando essa matriz com a matriz teste (**2**).

Criando um ciclo for com o número de amostras, colocando a rede a funcionar inserindo os dados da matriz teste e extraindo-os para a matriz *yplot* (**3**). No ponto **4**, definiu-se o limite de classificação que foi 0.9, colocando todos os dados arredondados na matriz y_bin.

Construindo um outro ciclo for (**5**), onde é comparado é colocado no gráfico cada ponto com uma respetiva cor da sua respetiva cor, sendo azul da classe A, vermelho da classe B, preto da classe C e roxo quando não pertencer a nenhuma destas classes.

5.1. Resultados obtidos

Nesta simulação com 10000 épocas, as matrizes dos pesos que são inicializadas aleatoriamente, ficaram com o valor inicial de:

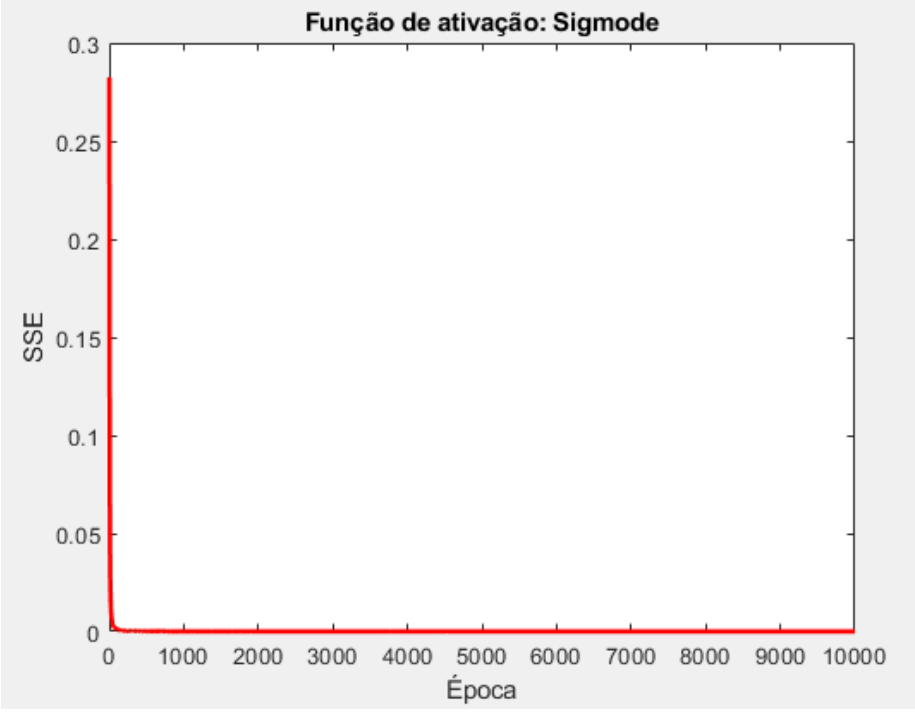
$$W1 = \begin{matrix} 0.3074 & -0.5190 & -0.4874 \\ 0.9139 & 0.4278 & 0.7882 \\ 0.8715 & 9.5187 & 0.3631 \\ -0.0842 & 0.4813 & -0.0735 \end{matrix}$$

$$W2 = \begin{matrix} -0.5757 & -0.6500 & 0.7888 & -0.6928 & 0.3595 \\ -0.8030 & -0.6729 & 0.0331 & 0.9069 & -0.9269 \\ 0.6471 & 0.3320 & 0.4054 & 0.0818 & 0.6184 \end{matrix}$$

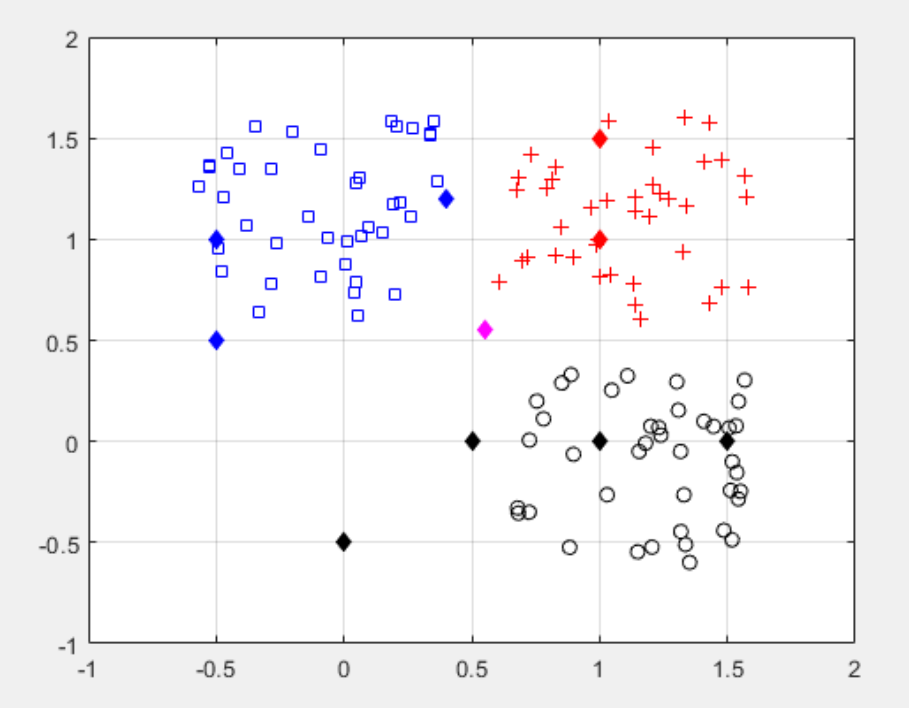
No final do treinamento obtivemos os seguintes resultados para os pesos:

$$W1 = \begin{matrix} 0.6133 & -11.1192 & 4.4866 \\ 11.0944 & -2.3732 & -2.6472 \\ -7.4232 & 2.3802 & 0.7711 \\ -0.3665 & 8.2191 & -4.1936 \end{matrix}$$

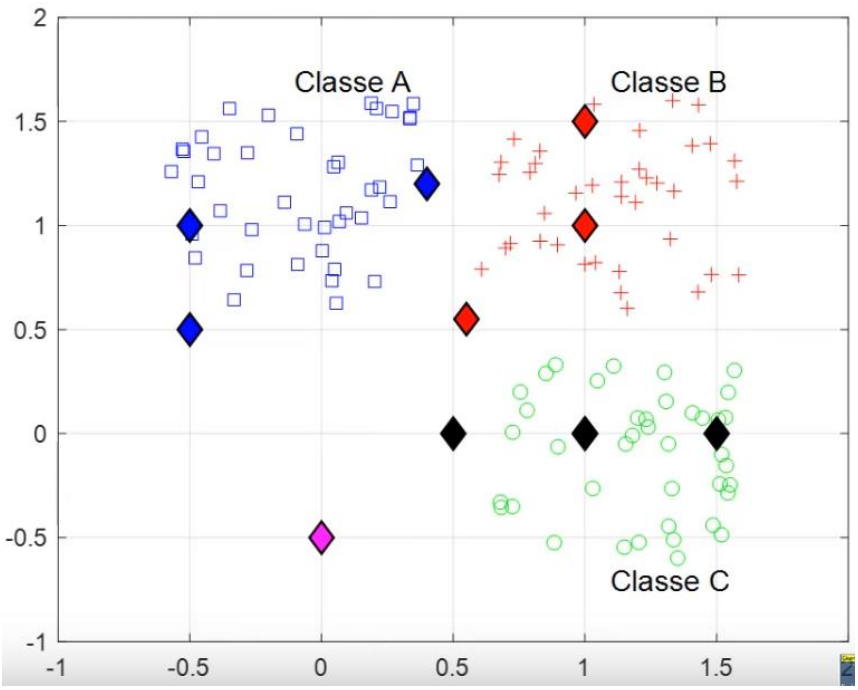
$$W2 = \begin{matrix} -1.4540 & -10.1445 & 5.8556 & 0.1678 & 2.0167 \\ -11.2302 & 8.8491 & -6.9431 & 5.0857 & -5.9606 \\ 8.9690 & 1.0277 & -2.2157 & -7.9328 & -1.2582 \end{matrix}$$



Assim concluímos que o treinamento da rede foi bem-sucedido visto existiu uma mudança nos valores do peso e o erro foi mínimo muito próximo a zero ao longo das épocas. Utilizando a matriz teste, o treino resultou os seguintes resultados:



Sendo os pontos em forma de losango são relativos à matriz teste, houve duas diferenças em relação ao gráfico desejado visto que a figura a seguir era o resultado desejado:



Comparando os dois gráficos percebemos que o ponto A= (0, 0.5) e o ponto B= (0.55,0.55) são de classes diferentes aquela desejada sendo o primeiro pertencente a nenhuma classe e o segundo à classe B.

A figura a seguir mostra o resultado de cada um dos dois pontos diferentes, sendo o Y o resultado atingido e o T o resultado expectável:

$$Y\_A = \begin{matrix}0.014\\0.8719\\0.0251\end{matrix} \quad \Rightarrow \quad Y\_bin\_A = \begin{matrix}0\\0\\0\end{matrix} \qquad Y\_A\_T = \begin{matrix}0\\1\\0\end{matrix}$$

Aproximando o valor segundo o limite de classificação

Assim podemos concluir que o 0.8719 não arredondou por conta de este ser menor do que o limite para a classificação, colocando assim o resultado a uma matriz que não se classifica em nenhuma classe apresentada, tornado assim o ponto roxo.

$$Y\_B = \begin{matrix}0.7261\\0.0000\\0.9991\end{matrix} \quad \Rightarrow \quad Y\_bin\_B = \begin{matrix}0\\0\\1\end{matrix} \qquad Y\_B\_T = \begin{matrix}1\\0\\1\end{matrix}$$

Aproximando o valor segundo o limite de classificação

O mesmo acontece no ponto B o que aconteceu no ponto A, era para o 0.7261 arredondar para 1 e assim este não pertencer a nenhuma classe, mas isso não aconteceu, o que causou com que o ponto B pertence-se assim à classe C.

Conclusão

Concluindo este relatório pode admitir-se que foi atingido o objetivo inicial proposto no relatório. Neste projeto aprendemos no software Matlab a conhecer melhor as redes neurais. Em relação à classificação de dados não linearmente separáveis posso concluir que com um maior número de eventos o erro diminui apresentando poucas falhas no seu resultado. Quanto à classificação multi-classe apesar de ter conseguido implementar o teste da rede, o resultado não atingiu o resultado esperado, mas consegue-se resolver diminuindo o limite para a classificação, apresentando mais dados no treino da rede e aumentando o número de épocas.

Bibliografia

Phil, K. (2017). Matlab deep learning with machine learning, neural networks and artificial intelligence. *Apress, New York*.

Paulo Moura Oliveira (2022), Tutorial sobre Redes Neuronais Artificiais- Classificação Multi-classe no Matlab (Em Português).
Youtube:*https://www.youtube.com/watch?v=8XeRqRcYN4s&t=582s*

Paulo Moura Oliveira (2022), Tutorial sobre Redes Neuronais Artificiais- Classificação do XOR no Matlab (Em Português) Youtube:
https://www.youtube.com/watch?v=U4bHTJJHZZQ