UNIVERSIDADE FEDERAL DO CEARÁ

CURSO: ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA: INTELIGÊNCIA COMPUTACIONAL

PROFESSOR: JARBAS JOACI DE MESQUITA SÁ JUNIOR

RELATÓRIO

Lucas Rodrigues da Silva - 428787

SOBRAL - CE

2021.2

1. REDE NEURAL MLP

O código referente à rede neural MLP encontra-se no arquivo rede_mlp.sce, ao qual foi-se utilizado o software Scilab versão 6.1.0. Assim, implementou-se uma rede MLP, a fim de classificar a base de dados "dermatology.data". Primeiramente, a base de dados foi carregada e as linhas com dados faltantes foram removidas, como mostra a figura 1 abaixo:

```
//pegando os dados da base dermatology.
base_original = csvRead('dermatology.data');

/*
* Retirando a linha dos dados faltantes
*/

r = isnan(base_original(:,34));
[t u] = size(base_original);

base = [];
cont = 1
for i=1:t
    if r(i) == %F
        base(cont,:) = base_original(i,:);
        cont = cont + 1;
    end
end
```

Figura 1 – Pegando a base de dados e retirando a linha dos dados faltantes

Assim, utilizou-se a função “**gsort()**”, própria do scilab, para organizar a base em ordem crescente de acordo com os valores de rótulos, que fica na coluna 35 da base, como mostra a figura 2.

```
/*
* Extraindo as amostras em vetores de entrada e vetores de rótulos
*/
[l m] = size(base);
// Ordenando as classes em ordem crescente de valores
[s,k] = (gsort(base(:,35),'lr','i'));

base_final = [];
for i=1:l
    base_final(i,:) = base(k(i),:);
end

x = base_final(:,1:34)'; // Vetores de entrada
// Pegando as classes e transformando na notação [1 0 0 0 0 0]
y = zeros(l,6);
for k=1:l
    aux = base_final(k,35);
    y(k,aux) = 1;
end
y = y'; // Vetores de rotulos
```

Figura 2 – Extraindo as amostras em matrizes de entrada e de rótulo.

A base foi separada em matriz de entrada e matriz de rótulos. Como a coluna 35 (classes da base) contém valores inteiros de 1 a 6, então utilizou-se a notação **[1 0 0 0 0 0]**, assim melhorando a classificação dos valores na rede neural. Logo após os dados estarem separados, utilizou-se a normalização por z-score na matriz de entrada. O objetivo desta normalização é alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores.

```
// Normalizando os valores de x por z-score
for i=1:34
    x(i,:) = (x(i,:) - mean(x(i,:)))/ stdev(x(i,:));
end
```

Figura 3 – Normalizando os valores de entrada por z-score.

A figura 4 abaixo mostra a separação das amostras em treino e teste, para a estratégia de validação, utilizou-se 50% das amostras de cada classe e separou-as em quatro (4) variáveis, como a base original continha 366 linhas e retirou-se 8 linhas que continham dados faltantes, então a base corrigida passou a ter 358 linhas somente. Assim a separação das amostras é mostrada na tabela 1:

```
/*
* Separando os valores em treino e teste
*/

//escolhendo os índices das amostras para treino e teste, 50% de cada classe.
index_treino = [1:55,112:142,172:206,243:266,291:314,339:348];
index_teste = [56:111,143:171,207:242,267:290,315:338,349:358];

x_treino = x(:,index_treino);
y_treino = y(:,index_treino);

x_teste = x(:,index_teste);
y_teste = y(:,index_teste);
```

Figura 4 – Separando as amostras em treino e teste.

Tabela 1 - Dados separados

Variável	Quantidade de amostras	% das amostras
x_treino e y_treino	179 amostras	50%
x_teste e y_teste	179 amostras	50%

A figura 6 mostra a função chamada de ***train_prev***, ela é utilizada para treinar os valores das matrizes ***x_treino*** e ***y_treino*** e também testar os valores obtidos com a matriz ***x_teste***.

A função ***train_prev*** tem como parâmetro de entrada um inteiro, que deve ser a quantidade de neurônios ocultos. Para treinar os valores das matrizes, utilizou-se a função ***ann_FFBP_gd***, que usa o algoritmo de backpropagation para treinar a rede MLP.

O algoritmo backpropagation, ou retropropagação dos erros, usa o gradiente descendente para calcular os erros dos neurônios ocultos fazendo uma projeção no sentido inverso ao do fluxo de informação convencional. A figura 5 abaixo detalha de forma simplificada esta projeção:

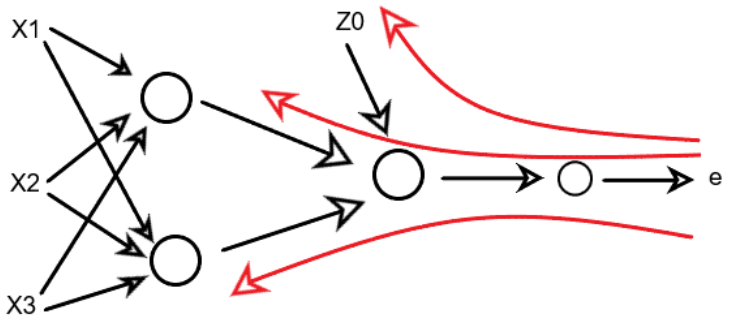

Figura 5 – Projeção da retropropagação dos erros.

A função ***ann_FFBP_gd*** recebe como parâmetro as matrizes de treino (***x_treino*** e ***y_treino*** neste caso), o número das unidades de entrada, a quantidade de neurônios ocultos e a quantidade de neurônios de saída.

Logo após o treino, é hora de testar os dados utilizando os valores obtidos no treinamento da rede MLP. Assim, utilizou-se a função ***ann_FFBP_run*** para testar os valores de treino obtidos anteriormente com a matriz de teste (***x_teste*** neste caso), com isso a função ***train_prev*** retorna esses valores de teste obtidos.

```
/*
* Treinando e testando a rede
*/

// Treino e teste
function [C]=train_prev(n)
    n_entrada = 34; // número de vetores de entrada
    n_rotulos = 6; // número de vetores de rótulos

    // ANN FeedForward Backpropagation Gradient Decent training function.
    W = ann_FFBP_gd(x_treino,y_treino,[n_entrada n n_rotulos]);

    // Prevendo o W com os dados de x
    C = ann_FFBP_run(x_teste,W);
endfunction
```

Figura 6 – Treino e Teste da rede MLP.

A figura 7 mostra a função ***calcular_acertos***, onde se calcula a quantidade de acertos da rede, utilizando uma matriz de valores de teste (***y_teste***).

O cálculo é feito pegando o máximo de todas as linhas de cada coluna da matriz de teste da base de dados e o máximo de todas as linhas de cada coluna dos valores obtidos da função ***ann_FFBP_run*** (figura 6). Assim que é obtido esse valor máximo, faz-se uma verificação, se o índice desses máximos são iguais, no caso de ser verdadeiro então contabiliza-se como acerto, no caso de ser falso então é um erro. Assim que os acertos são contabilizados a função ***calcular_acertos*** retorna a quantidade de acertos da rede MLP.

```
// Calculando a quantidade de acertos
function [cont]= calcular_acertos(y_teste, C)
    cont=0;
    for i=1:(l/2)
        [a b] = max(y_teste(:,i));
        [c d] = max(C(:,i));
        if b == d
            cont = cont + 1;
        end
    end
endfunction
```

Figura 7 – Verificando a quantidade de acertos da rede MLP.

A figura 8 mostra as chamadas das funções ***train_prev*** (figura 6) e da função ***calcular_acertos*** (figura 7), utilizando três valores distintos para a quantidade de neurônios ocultos, sendo eles os valores 50, 100, 175.

Para saber a percentagem de acertos da rede MLP, utilizou-se o valor de retorno da função ***calcular_acertos*** e dividiu-se este valor pelo número da quantidade de amostras utilizada para teste (que neste caso seria 179, como é observado na tabela 1).

```
// Número de neurônios na camada oculta
n_neuronios = [50 100 175];

for i=1:3
    C = train_prev(n_neuronios(i)); // treinamento e previsão
    cont(i) = calcular_acertos(y_teste,C); // Número de acertos
end

//Acertos da rede MLP.
for i=1:3
    result = cont(i)/(1/2);
    disp('---------- REDE MLP COM ' + string(n_neuronios(i)) + ' NEURÔNIOS OCULTOS ----------');
    disp('A percentagem de acertos da rede MLP com ' + string(n_neuronios(i)) + ' neurônios ocultos foi de: ' + string(result));
end
```

Figura 8 – Chamada de funções e resultado final da rede MLP.

Utilizando os valores 50, 100 e 175 para a quantidade de neurônios ocultos, obteve-se os seguintes resultados:

Tabela 2 - Percentagem de acertos de acordo com a quantidade de neurônios ocultos

Quantidade de Neurônios ocultos	Percentagem de acertos
50	0.58659
100	0.79330
175	0.82682

2. REDE NEURAL RBF

O código referente à rede neural RBF encontra-se no arquivo rede_rbf.sce. Assim, implementou-se uma rede RBF, a fim de classificar a base de dados "dermatology.data". Foram retirados os dados faltantes e a base foi separada em treino e teste, igualmente mostrado na rede neural MLP.

A figura 10 mostra a função utilizada para treinar a rede RBF com os dados de treino. Primeiramente, calculou-se a distância (norma) do vetor de entrada dos dados de treino ao centro de um cluster. Além disso, as saídas de cada neurônio da camada oculta foram definidas por meio de um função de ativação de base radial e ao resultado foi adicionado um bias igual a +1.

```
// Função que treina a rede
function [W] = treino(x_treino,y_treino, T, q, N, sigma)
    Z_treino = zeros(q, (N/2));

    for i=1: (N/2)
        for j=1:q
            v = norm(x_treino(:,i) - T(:,j));
            Z_treino(j,i) = exp(-v^2/2*(sigma^2));
        end
    end

    Z_treino = [ones(1, (N/2));Z_treino];

    W = (y_treino*Z_treino') * ((Z_treino*Z_treino')^(-1));

endfunction
```

Figura 9 – Função de treino da rede RBF.

Com isso, para definir os pesos da camada de saída, utilizou-se a equação:

$$D = W * Z \tag{1}$$

onde,

- D é a matriz de rótulos;
- W é a matriz dos pesos da camada de saída;
- Z é a matriz dos vetores produzidos pela camada oculta.

Assim, utilizando a pseudo inversa, para descobrir a matriz dos pesos, chegou-se a seguinte equação:

$$D * Z^T * (Z * Z^T)^{-1} = W * (Z * Z^T) * (Z * Z^T)^{-1} \quad (2)$$

Logo, como o resultado de $(Z * Z^T)$ é uma matriz quadrada e esta matriz multiplicada pela sua inversa é a matriz identidade, então como resultado final obteve-se a equação da matriz de pesos da camada de saída, explicitada abaixo:

$$W = D * Z^T * (Z * Z^T)^{-1} \quad (3)$$

Em seguida, utilizou-se o mesmo método da figura 9 para calcular a distância dos dados de teste com os centróides, adicionando o bias igual a +1 ao resultado. Logo após este cálculo, multiplicou-se os valores obtidos com os resultados da equação (3). Com isso, obteve-se o resultado de previsão da rede RBF.

Dessa maneira, a rede RBF utilizou a mesma forma da rede MLP de descobrir a quantidade de acertos. Pegando os valores máximos e comparando seus índices.

```
//Função que testa a rede
function [cont] = teste(x_teste,y_teste,W,T,q,N, sigma)
    Z_teste = zeros(q,(N/2));

    for i=1:(N/2)
        for j=1:q
            v = norm(x_teste(:,i) - T(:,j));
            Z_teste(j,i) = exp(-v^2/2*(sigma^2));
        end
    end

    Z_teste = [ones(1,(N/2));Z_teste];
    prev = W * Z_teste;

    cont = 0;
    for k=1:(N/2)
        [a b] = max(prev(:,k));
        [c d] = max(y_teste(:,k));
        if b == d
            cont = cont + 1;
        end
    end
endfunction
```

Figura 10 – Função de treino da rede RBF.

Por fim, pegou-se os valores de retorno da função ***teste*** e dividiu os pela quantidade de amostras utilizadas no teste (tabela 1), obtendo assim a percentagem de acertos da rede neural RBF.

A figura 11 mostra a chamada da função ***treino*** e da função ***teste***. Utilizou-se três valores distintos para a quantidade de neurônios ocultos, sendo eles 2, 10 e 25. Com isso, pode-se saber qual a melhor quantidade de neurônios para esta rede neural.

```
[p N] = size(x);
sigma = 0.3;
// p é a quantidade de atributos em cada valor de entrada x
q = [2 10 25]; ///Quantidade de neurônios ocultos

//Testando e Treinando a rede com as amostras.
for i=1:3
    T = rand(p,q(i),'normal'); //Vetores centroide dos q neurônios
    W = treino(x_treino,y_treino,T,q(i),N,sigma);
    cont(i) = teste(x_teste,y_teste,W,T,q(i),N, sigma);
end

//Acertos da rede RBF.
for i=1:3
    result = cont(i)/(N/2);
    disp('---------REDE RBF COM ' + string(q(i)) + ' NEURÔNIOS OCULTOS ----------');
    disp('A percentagem da rede RBF com ' + string(q(i)) + ' neurônios ocultos foi de: ' + string(result));
end
```

Figura 11 – Chamada de funções e resultado final da rede RBF

Portanto, a tabela 3 apresenta os resultados de acerto da rede neural RBF para diferentes quantidades de neurônios ocultos.

Tabela 3 - Resultados da rede para diferentes quantidade de neurônios ocultos.

Quantidade de neurônios ocultos	Resultado
2	0.39106
10	0.83799
25	0.9162

Como os centróides sempre mudam ao executar o código da rede, então os valores sempre irão oscilar um pouco, mas nunca estarão muito distantes dos valores de resultado obtidos na tabela 3.

3. SAÍDA DO CÓDIGO PARA CADA REDE NEURAL

```
"--------- REDE MLP COM 50 NEURÔNIOS OCULTOS -------------"
"A percentagem de acertos da rede MLP com 50 neurônios ocultos foi de: 0.56983"
"--------- REDE MLP COM 100 NEURÔNIOS OCULTOS -------------"
"A percentagem de acertos da rede MLP com 100 neurônios ocultos foi de: 0.77654"
"--------- REDE MLP COM 175 NEURÔNIOS OCULTOS -------------"
"A percentagem de acertos da rede MLP com 175 neurônios ocultos foi de: 0.85475"
```

Figura 12 – Resultado da rede neural MLP

```
"--------- REDE RBF COM 2 NEURÔNIOS OCULTOS -------------"
"A percentagem da rede RBF com 2 neurônios ocultos foi de: 0.48045"
"--------- REDE RBF COM 10 NEURÔNIOS OCULTOS -------------"
"A percentagem da rede RBF com 10 neurônios ocultos foi de: 0.82682"
"--------- REDE RBF COM 25 NEURÔNIOS OCULTOS -------------"
"A percentagem da rede RBF com 25 neurônios ocultos foi de: 0.94413"
```

Figura 13 – Resultado da rede neural RBF