Prova 1 - Inteligência Artificial

Henrique Sander Lourenço 10802705

4 de outubro de 2022

1 Entradas

Para treinar a rede neural, utilizou-se uma matriz X, em que cada linha corresponde a uma entrada (um estado), de acordo com seus critérios de viabilidade, 1 indicando viável (V) e 0 indicando não viável (NV). A ordem dos critérios corresponde à ordem enunciada: IE, VF, EB, PI, D1, D2, D3 e D5. Sendo assim, X ficou conforme mostrado no seguinte trecho de código:

```
% Entradas: 3 vetores de 16 posicoes indicando os criterios de
    viabilidade
% 1: Viavel (V)
% 0: Nao viavel (NV)
X = [1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0; % SC
     1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0; % RJ
     1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1]; % SP
```

2 Saídas

As saídas foram representadas pela matriz y, sendo que cada linha corresponde à saída correspondente para determinada entrada. Cada saída é composta por um vetor de números zeros (0) com um número um (1) indicando qual estado foi identificado. Portanto, as saídas desejadas são representadas pela seguinte matriz D:

```
% Saidas:
% 1: indica o estado correspondente
% 1a posicao: SC
% 2a posicao: RJ
% 3a posicao: SP
D = [1 0 0;
     0 1 0;
     0 0 1];
```

3 Pesos

Sendo a soma do número USP do aluno 23, foram utilizados 23 nós na camada escondida. Visto que o tamanho de cada entrada é 16, para a primeira matriz

de peso, denominada W_1, foi usada uma matriz de dimensão 23x16. Já quanto à matriz da camada de saída, denominada W_2, foi utilizada uma matriz de dimensão 3x23 (tamanho da saída x quantidade de nós da camada escondida). O seguinte trecho de código ilustra a inicialização dos pesos com cada elemento situando-se entre -1 e 1:

```
% numero de nos da camada escondida x tamanho das entradas
W1 = 2 * rand(23, 16) - 1;
% tamanho das saidas x numero de nos da camada escondida
W2 = 2 * rand(3, 23) - 1;
```

4 Treinamento

O treinamento foi feito utilizando-se o método do gradiente descendente estocástico (GDE) com backpropagation, em que: o sinal de entrada é propagado naturalmente até a saída, utilizando funções de ativação sigmoide; os erros (e) são calculados a partir desta, sabendo a saída desejada; o delta correspondente é calculado aplicando a derivada da função de função de ativação à soma ponderada do nó (v) e multiplicando o resultado pelo erro ($\delta = \phi'(v)e$); o erro na saída da camada escondida (e_1) é calculado multiplicando W_2 por δ; um novo delta δ_1, correspondente à camada escondida é calculado da mesma forma e, finalmente, os pesos são atualizados conforme:

$$dW_1 = \alpha\delta_1 x^{'}$$

$$dW_2 = \alpha\delta_1 y_1^{'}$$

em que α é a taxa de aprendizagem (utilizada 0.9) e y_1 a saída da camada escondida.

O seguinte trecho de código mostra o treinamento, em que é feito também o cálculo do erro quadrático mínimo:

```
% Treinamento da rede neural com gradiente descendente estocastico
    por backpropagation

function [W1, W2, meq] = BackpropSGD(W1, W2, X, D, numeroEntradas)
    alpha = 0.9; % Taxa de aprendizagem
    acc = 0; % Acumulador para calculo do erro quadr tico m nimo
    for k = 1:numeroEntradas
        x = X(k, :)'; % Entrada k da rede neural
        d = D(k, :)'; % Saida desejada para respectiva entrada
        v1 = W1 * x;
        y1 = sigmoid(v1); % Saida dos nos da camada escondida
        v  = W2 * y1;
        y = sigmoid(v); % Saida dos nos da camada de sa da
        e = d - y; % Erro na saida da rede neural
        acc = acc + min(e.^2); % Calculo do erro quadratico minimo
        delta = y .* (1 - y) .* e;
        e1 = W2' * delta; % Erro na saida dos nos da camada
            escondida
```

```
17          delta1 = y1 .* (1 - y1) .* e1;
18          % Atualizacao dos pesos
19          dW1 = alpha * delta1 * x';
20          W1 = W1 + dW1;
21          dW2 = alpha * delta * y1';
22          W2 = W2 + dW2;
23      end
24      meq = acc / numeroEntradas; % Erro quadratico minimo
25  end
```

5 Decisão

Por último, a inferência é feita e, baseando-se nas saídas obtidas, o programa decide o estado correspondente a cada entrada, com um algoritmo que retorna erro caso o estado não seja identificado corretamente:

```
1   % Inferencia
2   y = zeros(numeroSaidas, tamanhoSaida);
3   for k = 1:numeroEntradas
4       x = X(k, :)';
5       v1 = W1 * x;
6       y1 = sigmoid(v1);
7       v = W2 * y1;
8       y(k, :) = sigmoid(v);
9   end
10
11  % Para cada saida, indica se foi possivel identificar
12  % o estado e, se sim, qual estado foi identificado
13  for j = 1:3
14      z = y(j, :);
15      % Flag para saber se o estado foi identificado
16      % 1: nao identificado
17      % 0: identificado
18      estadoNaoIdentificado = 0;
19      for i = 1:3
20          % Se algum elemento da saida ficar entre 0.1 e 0.9,
21          % nao foi possivel identificar o estado
22          if z(i) < 0.9 && z(i) > 0.1
23              estadoNaoIdentificado = 1;
24              disp("N o foi poss vel identificar o estado");
25              break
26          end
27      end
28
29      % Se o estado foi identificado, verifica qual deles foi
30      if estadoNaoIdentificado == 0
31          if z(1) >= 0.9 && z(2) <= 0.1 && z(3) <= 0.1
32              disp("Estado: SC")
33          elseif z(2) >= 0.9 && z(1) <= 0.1 && z(3) <= 0.1
34              disp("Estado: RJ")
35          elseif z(3) >= 0.9 && z(1) <= 0.1 && z(2) <= 0.1
36              disp("Estado: SP")
37          % Caso nenhum ou mais de um elemento da saida apresente
                valor
38          % maior que 0.9, a identificacao falhou
```

```
39          else
40              disp("Erro desconhecido ao identificar a imagem")
41          end
42      end
43  end
```

6 Erro

O treinamento apresentou, em função da iteração do treinamento, o erro mostrado na figura 1.

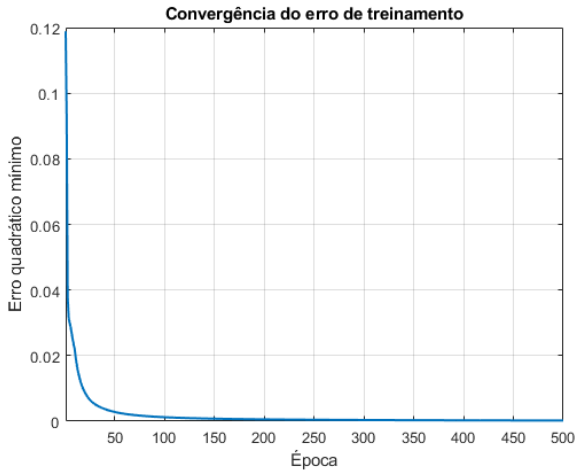


Figura 1: Erro quadrático mínimo

7 Saídas desejadas e obtidas

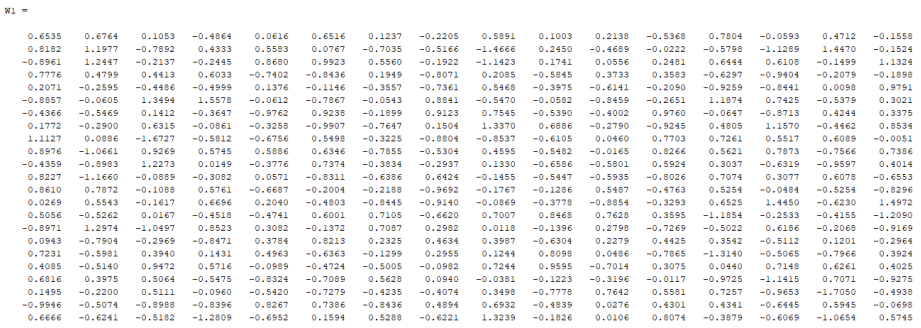
As saídas desejadas e obtidas são comparadas na figura 2.

```
Resultados:
                [saída desejada]   [saída obtida]
    1.0000         0         0    0.9762    0.0188    0.0190
         0    1.0000         0    0.0172    0.9783    0.0107
         0         0    1.0000    0.0205    0.0132    0.9782
```

Figura 2: Comparação entre saídas desejadas e obtidas

8 Pesos resultantes

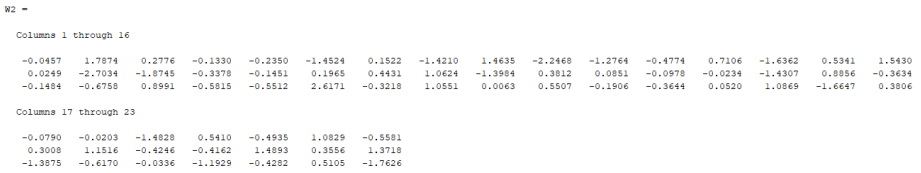
Sendo uma matriz de peso 23x16 e outra 3x23, são 437 pesos. As matrizes W_1 e W_2 resultantes são mostradas nas figuras **??** e **??**.



```
W1 =

    0.6535    0.6764    0.1053   -0.4864    0.0616    0.6516    0.1237   -0.2205    0.5891    0.1003    0.2138   -0.5368    0.7804   -0.0593    0.4712   -0.1558
    0.8182    1.1977   -0.7892    0.4333    0.5583    0.0767   -0.7035   -0.5166   -1.4666    0.2450   -0.4689   -0.0222   -0.5798   -1.1289    1.4470   -0.1524
   -0.8961    1.2447   -0.2137   -0.2445    0.8680    0.9923    0.5560   -0.1922   -1.1423    0.1741    0.0556    0.2481    0.6444    0.6108   -0.1499    1.1324
    0.7776    0.4799    0.4413    0.6033   -0.7402   -0.8436    0.1949   -0.8071    0.2085   -0.5845    0.3733    0.3583   -0.6297   -0.9404   -0.2079   -0.1898
    0.2071   -0.2595   -0.4486   -0.4999    0.1376   -0.1146   -0.3557   -0.7361    0.5468   -0.3975   -0.6141   -0.2090   -0.9259   -0.8441    0.0098    0.9791
   -0.8857   -0.0605    1.3494    1.5578   -0.0612   -0.7867   -0.0543    0.8841   -0.5470   -0.0582   -0.8459   -0.2651    1.1874    0.7425   -0.5379    0.3021
   -0.4366   -0.5469    0.1412   -0.3647   -0.9762    0.9238   -0.1899    0.9123    0.7545   -0.5390   -0.4002    0.9760   -0.0647   -0.8713    0.4244    0.3375
    0.1772   -0.2900    0.6315   -0.0861   -0.3258   -0.9907   -0.7647    0.1504    1.3370    0.6886   -0.2790   -0.9245    0.4805    1.1570   -0.4462    0.8534
    1.1127    0.0886   -1.6727   -0.5812   -0.6756    0.5498   -0.3225   -0.8804   -0.8537   -0.6105    0.0460    0.7703    0.7261    0.5517    0.6089   -0.0051
    0.8976   -1.0661    0.9269    0.5745    0.5886    0.6346   -0.7855   -0.5304    0.4595   -0.5482   -0.0165    0.8266    0.5621    0.7873   -0.7566    0.7386
   -0.4359   -0.8983    1.2273    0.0149   -0.3776    0.7374   -0.3834   -0.2937    0.1330   -0.6586   -0.5801    0.5924    0.3037   -0.6319   -0.9597    0.4014
    0.8227   -1.1660   -0.0889   -0.3082    0.0571   -0.8311   -0.6386    0.6424   -0.1455   -0.5447   -0.5935   -0.8026    0.7074    0.3077    0.6078   -0.6553
    0.8610    0.7872   -0.1088    0.5761   -0.6687   -0.2004   -0.2188   -0.9692   -0.1767   -0.1286    0.5487   -0.4763    0.5254   -0.0484   -0.5254   -0.8296
    0.0269    0.5543   -0.1617    0.6696    0.2040   -0.4803   -0.8445   -0.9140   -0.0869   -0.3778   -0.8854   -0.3293    0.6525    1.4450   -0.6230    1.4972
    0.5056   -0.5262    0.0167   -0.4518   -0.4741    0.6001    0.7105   -0.6620    0.7007    0.8468    0.7628    0.3595   -1.1854   -0.2533   -0.4155   -1.2090
   -0.8971    1.2974   -1.0497    0.8523    0.3082   -0.1372    0.7087    0.2982    0.0118   -0.1396    0.2798   -0.7269   -0.5022    0.6186   -0.2068   -0.9169
    0.0943   -0.7904   -0.2969   -0.8471    0.3784    0.8213    0.2325    0.4634    0.3987   -0.6304    0.2279    0.4425    0.3542   -0.5112    0.1201   -0.2964
    0.7231   -0.5981    0.3940    0.1431    0.4963   -0.6363   -0.1299    0.2955    0.1244    0.8098    0.0486   -0.7865   -1.3140   -0.5065   -0.7966    0.3924
    0.4085   -0.5140    0.9472    0.5716   -0.0989   -0.4724   -0.5005   -0.0982    0.7244    0.9595   -0.7014    0.3075    0.0440    0.7148    0.6261    0.4025
    0.6816    0.3975    0.5064   -0.5475   -0.8324   -0.7089    0.5628    0.0940   -0.0381   -0.1223   -0.3196   -0.0117   -0.9725   -1.1415    0.7071   -0.9275
    0.1495   -0.2200    0.5111   -0.0960   -0.5420   -0.7279   -0.4235   -0.4074    0.3498   -0.7778    0.7642    0.5581    0.7257   -0.9653   -1.7050   -0.4938
   -0.9946   -0.5074   -0.8988   -0.8396    0.8267    0.7386   -0.8436    0.4894    0.6932   -0.4839    0.0276    0.4301    0.4341   -0.6445    0.5945   -0.0698
    0.6666   -0.6241   -0.5182   -1.2809   -0.6952    0.1594    0.5288   -0.6221    1.3239   -0.1826    0.0106    0.8074   -0.3879   -0.6069   -1.0654    0.5745
```

Figura 3: Matriz W_1



```
W2 =

  Columns 1 through 16

   -0.0457    1.7874    0.2776   -0.1330   -0.2350   -1.4524    0.1522   -1.4210    1.4635   -2.2468   -1.2764   -0.4774    0.7106   -1.6362    0.5341    1.5430
    0.0249   -2.7034   -1.8745   -0.3378   -0.1451    0.1965    0.4431    1.0624   -1.3984    0.3812    0.0851   -0.0978   -0.0234   -1.4307    0.8856   -0.3634
   -0.1484   -0.6758    0.8991   -0.5815   -0.5512    2.6171   -0.3218    1.0551    0.0063    0.5507   -0.1906   -0.3644    0.0520    1.0869   -1.6647    0.3806

  Columns 17 through 23

   -0.0790   -0.0203   -1.4828    0.5410   -0.4935    1.0829   -0.5581
    0.3008    1.1516   -0.4246   -0.4162    1.4893    0.3556    1.3718
   -1.3875   -0.6170   -0.0336   -1.1929   -0.4282    0.5105   -1.7626
```

Figura 4: Matriz W_2

9 Conclusão

A curva do erro quadrático mínimo mostra a rápida convergência do treinamento. As saídas obtidas apresentaram um erro muito pequeno e o programa foi capaz de identificar cada estado, mostrando que o treinamento foi bem sucedido.

10 Anexos

10.1 Código principal

```
% SEL0362 - Intelig ncia Artificial
% Prova 1
% Henrique Sander Louren o - 10802705

clear all
close all
clc

% Entradas: 3 vetores de 16 posicoes indicando os criterios de
    viabilidade
% 1: Viavel (V)
% 0: Nao viavel (NV)
X = [1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0; % SC
     1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0; % RJ
     1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1]; % SP

% Saidas:
% 1: indica o estado correspondente
% 1a posicao: SC
% 2a posicao: RJ
% 3a posicao: SP
D = [1 0 0;
     0 1 0;
     0 0 1];

somaNusp = 23; % Soma NUSP (10802705)
numeroEntradas = 3; % Numero de entradas
tamanhoEntrada = 16; % Tamanho do vetor de entrada
numeroSaidas = 3; % Numero de saidas
tamanhoSaida = 3; % Tamanho do vetor de saida

% Inicializacao dos pesos da camada escondida
% numero de nos da camada escondida x tamanho das entradas
W1 = 2 * rand(somaNusp, tamanhoEntrada) - 1;

% Inicializacao dos pesos da camada de saida
% tamanho das saidas x numero de nos da camada escondida
W2 = 2 * rand(tamanhoSaida, somaNusp) - 1;

% Treinamento por back-propagation SGD:
epoca = 500;
meq = zeros(1, epoca); % Inicializacao do erro quadratico minimo
for i = 1:epoca
    [W1, W2, meq(i)] = BackpropSGD(W1, W2, X, D, numeroEntradas);
end

% Inferencia
y = zeros(numeroSaidas, tamanhoSaida);
for k = 1:numeroEntradas
    x = X(k, :)';
    v1 = W1 * x;
    y1 = sigmoid(v1);
    v = W2 * y1;
```

```
53      y(k, :) = sigmoid(v);
54  end
55
56  disp('Resultados:');
57  disp('               [sa da desejada]  [sa da obtida]');
58  disp([D y])
59
60  % Para cada saida, indica se foi possivel identificar
61  % o estado e, se sim, qual estado foi identificado
62  for j = 1:3
63      z = y(j, :);
64      % Flag para saber se o estado foi identificado
65      % 1: nao identificado
66      % 0: identificado
67      estadoNaoIdentificado = 0;
68      for i = 1:3
69          % Se algum elemento da saida ficar entre 0.1 e 0.9,
70          % nao foi possivel identificar o estado
71          if z(i) < 0.9 && z(i) > 0.1
72              estadoNaoIdentificado = 1;
73              disp("N o foi poss vel identificar o estado");
74              break
75          end
76      end
77
78      % Se o estado foi identificado, verifica qual deles foi
79      if estadoNaoIdentificado == 0
80          if z(1) >= 0.9 && z(2) <= 0.1 && z(3) <= 0.1
81              disp("Estado: SC")
82          elseif z(2) >= 0.9 && z(1) <= 0.1 && z(3) <= 0.1
83              disp("Estado: RJ")
84          elseif z(3) >= 0.9 && z(1) <= 0.1 && z(2) <= 0.1
85              disp("Estado: SP")
86          % Caso nenhum ou mais de um elemento da saida apresente
                valor
87          % maior que 0.9, a identificacao falhou
88          else
89              disp("Erro desconhecido ao identificar a imagem")
90          end
91      end
92  end
93
94  figure(1)
95  plot(1:1:epoca, meq, 'LineWidth', 1.5);
96  title('Converg ncia do erro de treinamento');
97  xlabel(' poca ');
98  ylabel('Erro quadr tico m nimo');
99  xlim([1 epoca]);
100 grid on;
```

10.2 GDE por backpropagation

```
1   % Treinamento da rede neural com gradiente descendente estocastico
        por backpropagation
2
3   function [W1, W2, meq] = BackpropSGD(W1, W2, X, D, numeroEntradas)
```

```
4       alpha = 0.9; % Taxa de aprendizagem
5       acc = 0; % Acumulador para calculo do erro quadr tico m nimo
6       for k = 1:numeroEntradas
7           x = X(k, :)'; % Entrada k da rede neural
8           d = D(k, :)'; % Saida desejada para respectiva entrada
9           v1 = W1 * x;
10          y1 = sigmoid(v1); % Saida dos nos da camada escondida
11          v  = W2 * y1;
12          y = sigmoid(v); % Saida dos nos da camada de sa da
13          e = d - y; % Erro na saida da rede neural
14          acc = acc + min(e.^2); % Calculo do erro quadratico minimo
15          delta = y .* (1 - y) .* e;
16          e1 = W2' * delta; % Erro na saida dos nos da camada
                escondida
17          delta1 = y1 .* (1 - y1) .* e1;
18          % Atualizacao dos pesos
19          dW1 = alpha * delta1 * x';
20          W1 = W1 + dW1;
21          dW2 = alpha * delta * y1';
22          W2 = W2 + dW2;
23      end
24      meq = acc / numeroEntradas; % Erro quadratico minimo
25  end
```

10.3 Função sigmoide

```
1   % Definicao da fun ao sigmoide
2   function sigmoid = sigmoid(x)
3        sigmoid = 1 ./ (1 + exp(-x));
4   end
```