Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

Alunos: Luisa Helena Bartocci Liboni
Rodrigo de Toledo Caropreso

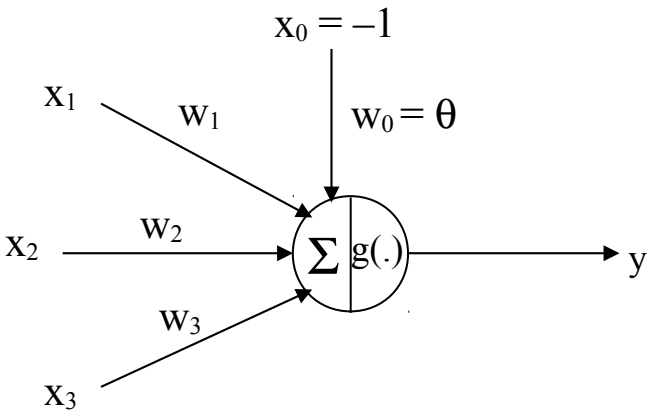
Redes Neurais Artificiais

EPC-1

A partir da análise de um processo de destilação fracionada de petróleo observou-se que determinado óleo poderia ser classificado em duas classes de pureza {C1 e C2}, mediante a medição de três grandezas {x_1, x_2 e x_3} que representam algumas das propriedades físico-químicas do óleo. A equipe de engenheiros e cientistas pretende utilizar um perceptron para executar a classificação automática destas duas classes.

Assim, baseadas nas informações coletadas do processo, formou-se o conjunto de treinamento em anexo, tomando por convenção o valor –1 para óleo pertencente à classe C1 e o valor +1 para óleo pertencente à classe C2.

Portanto, o neurônio constituinte do perceptron terá três entradas e uma saída, conforme ilustrado na figura abaixo:



Utilizando o algoritmo supervisionado de Hebb (regra de Hebb) para classificação de padrões, e assumindo-se a taxa de aprendizagem igual a 0.01, faça as seguintes atividades:

1. Execute 5 treinamentos para a rede perceptron, inicializando-se o vetor de pesos em cada treinamento com valores aleatórios entre zero e um. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos do vetor de pesos iniciais não sejam os mesmos.

2. Registre os resultados dos 5 treinamentos na tabela seguinte:

Treino	Vetor de Pesos Inicial				Vetor de Pesos Final				Número de Épocas
	w_0	w_1	w_2	w_3	w_0	w_1	w_2	w_3	
1º (T1)	0.5657	0.7165	0.5113	0.7764	-0.0143	0.2656	0.1457	0.3264	105
2º (T2)	0.4893	0.1859	0.7006	0.9827	-0.0107	0.1952	0.1070	-0.2398	114
3º (T3)	0.8066	0.7036	0.4850	0.1146	0.0134	0.3981	0.2260	-0.5031	16
4º (T4)	0.6649	0.3654	0.1400	0.5668	-0.0151	0.2682	0.1489	-0.3282	152
5º (T5)	0.8230	0.6739	0.9994	0.9616	0.0030	0.2617	0.1488	-0.3307	24

3. Após o treinamento do perceptron, aplique então o mesmo na classificação automática de novas amostras de óleo, indicando-se na tabela seguinte os resultados das saídas (Classes) referentes aos cinco processos de treinamento realizados no item 1.

Amostra	x_1	x_2	x_3	y (T_1)	y (T_2)	y (T_3)	y (T_4)	y (T_5)
1	-0.3565	0.0620	5.9891	-1	-1	-1	-1	-1
2	-0.7842	1.1267	5.5912	+1	+1	+1	+1	+1
3	0.3012	0.5611	5.8234	+1	+1	+1	+1	+1
4	0.7757	1.0648	8.0677	+1	+1	+1	+1	+1
5	0.1570	0.8028	6.3040	+1	+1	+1	+1	+1
6	-0.7014	1.0316	3.6005	+1	+1	+1	+1	+1
7	0.3748	0.1536	6.1537	+1	+1	-1	+1	-1
8	-0.6920	0.9404	4.4058	+1	+1	+1	+1	+1
9	-1.3970	0.7141	4.9263	-1	-1	-1	-1	-1
10	-1.8842	-0.2805	1.2548	-1	-1	-1	-1	-1

4. Explique por que o número de épocas de treinamento varia a cada vez que se executa o treinamento do perceptron.

O número de épocas varia porque a inicialização dos pesos é diferente em cada treinamento, colocando o Perceptron em um estado inicial tal que pode demorar mais tempo ou menos tempo para que o treinamento completo seja alcançado. Isso se reflete em uma quantidade maior ou menor de épocas, dependendo do caso.

5. Qual a principal limitação do perceptron quando aplicado em problemas de classificação de padrões.

A principal limitação do perceptron é que ele somente é capaz de classificar conjuntos linearmente separáveis. Caso contrário, ele não irá realizar a classificação dos conjuntos corretamente, gerando uma quantidade de erros muito alta quando estiver em operação.

OBSERVAÇÕES:

1. O EPC pode ser realizado em grupo de três pessoas. Se for o caso, entregar somente um EPC com o nome de todos integrantes.

2. As folhas contendo os resultados do EPC devem ser entregue em sequência e grampeadas (não use clips).
3. Em se tratando de EPC que tenha implementação computacional, anexe (de forma impressa) o programa fonte referente ao mesmo.

ANEXO – Conjunto de Treinamento.

Amostra	x_1	x_2	x_3	d
01	-0.6508	0.1097	4.0009	-1.0000
02	-1.4492	0.8896	4.4005	-1.0000
03	2.0850	0.6876	12.0710	-1.0000
04	0.2626	1.1476	7.7985	1.0000
05	0.6418	1.0234	7.0427	1.0000
06	0.2569	0.6730	8.3265	-1.0000
07	1.1155	0.6043	7.4446	1.0000
08	0.0914	0.3399	7.0677	-1.0000
09	0.0121	0.5256	4.6316	1.0000
10	-0.0429	0.4660	5.4323	1.0000
11	0.4340	0.6870	8.2287	-1.0000
12	0.2735	1.0287	7.1934	1.0000
13	0.4839	0.4851	7.4850	-1.0000
14	0.4089	-0.1267	5.5019	-1.0000
15	1.4391	0.1614	8.5843	-1.0000
16	-0.9115	-0.1973	2.1962	-1.0000
17	0.3654	1.0475	7.4858	1.0000
18	0.2144	0.7515	7.1699	1.0000
19	0.2013	1.0014	6.5489	1.0000
20	0.6483	0.2183	5.8991	1.0000
21	-0.1147	0.2242	7.2435	-1.0000
22	-0.7970	0.8795	3.8762	1.0000
23	-1.0625	0.6366	2.4707	1.0000
24	0.5307	0.1285	5.6883	1.0000
25	-1.2200	0.7777	1.7252	1.0000
26	0.3957	0.1076	5.6623	-1.0000
27	-0.1013	0.5989	7.1812	-1.0000
28	2.4482	0.9455	11.2095	1.0000
29	2.0149	0.6192	10.9263	-1.0000
30	0.2012	0.2611	5.4631	1.0000

CÓDIGO FONTE UTILIZADO

Carregamento dos Dados

```
[DB_X1   DB_X2   DB_X3   DB_D]   =
textread(  'Dados_Treino.dat',  '%f
%f %f %f', 'headerlines', 1);

[DB_X1     DB_X2     DB_X3]     =
textread( 'Dados_Operacao.dat', '%f
%f %f', 'headerlines', 1);
```

Normalização

```
function        NormArray        =
Normaliza(  max_scale,  min_scale,
Vetor_Amostras )
%Normaliza Faz a normalizacao do
vetor de amostras
%  max_scale  ->  valor  maximo  da
escala (para funcao sgn = 1)
%  min_scale  ->  valor  minimo  da
escala (para funcao sgn = -1)
%  Vetor_Amostras  ->  vetor  com
amostras

max_amostras = max(Vetor_Amostras);
min_amostras = min(Vetor_Amostras);

for k=1:length(Vetor_Amostras)
      NormArray(k)  =  (max_scale  -
min_scale)  *  (  Vetor_Amostras(k)  -
min_amostras  )  /  (max_amostras  -
min_amostras) + min_scale;
end;
```

Treinamento do Perceptron

```
function pesos = Perceptron_Treino(
eta, entradas, saidas, max_epocas )
%Perceptron_Treino  Treinamento  de
Perceptron 1 camada
%   eta          -> coeficiente de
treino
%    entradas       -> matriz  com
entradas
%   saidas      -> vetor com saidas
desejadas
%   max_epocas  -> limite de epocas
de treinamento

sizeW = size(entradas);
N_entradas = sizeW(1);
N_amostras = sizeW(2);
pesos = rand(N_entradas, 1);

disp('Inicialização do Perceptron -
Pesos  (Pressione  uma  tecla  para
continuar)');
pesos
pause

%inicio do treinamento
epoca = 0;
erro = 1;

while(  epoca  <=  max_epocas  &&
erro )
    erro = 0;
    for k=1:N_amostras
        u = pesos' * entradas( :, k
);
        y = sign( u );

        %verifica erro
        if( y ~= saidas( k ) )
            %corrige peso
%             eta * ( saidas( k ) -
y ) * entradas( k )
            pesos = pesos + eta * (
saidas( k ) - y ) * entradas( :,
k );
            erro = 1;
        end
    end;
    epoca = epoca + 1;
end;

epoca = epoca - 1;

disp( sprintf( 'Fim do treinamento.
Numero de epocas: %d', epoca) );
```

Operação do Perceptron

```
function            y            =
Perceptron_Executa(          pesos,
entrada )
%Perceptron_Executa   Operacao   de
Perceptron 1 camada
%    entradas     -> vetor com uma
entrada
%   pesos        -> matriz de pesos
do treinamento
%   max_epocas  -> limite de epocas
de treinamento

u = pesos' * entrada;
y = sign( u );
```

```
if( y == -1 )
         disp( sprintf( 'Amostra
pertence a classe C1 (-1)') );
else
         disp( sprintf( 'Amostra
pertence a classe C2 (+1)') );
end;
```

Execução do EPC1

```
%Carrega os dados
Carrega_Tabela_Treino;

%Monta vetores de amostras
N_entradas = 3; %entradas do
perceptron
eta = 0.1; %coeficiente de
treinamento

%Normaliza dados (pre-
processamento)
DB_X1_Norm = Normaliza( 1, -1,
DB_X1 );
DB_X2_Norm = Normaliza( 1, -1,
DB_X2 );
DB_X3_Norm = Normaliza( 1, -1,
DB_X3 );

x = [];
%monta matriz de entradas
for k=1: length(DB_X1_Norm)
     x(:, k) = [ -1 DB_X1_Norm(k)
DB_X2_Norm(k) DB_X3_Norm(k) ]';
end;

d = [DB_D];
max_epocas = 1000;

pesos = Perceptron_Treino(eta, x,
d, max_epocas);

disp('Pesos do Perceptron
treinado');
pesos
pause

%OPERACAO
%Carrega os dados
Carrega_Tabela_Operacao;

%Monta vetores de amostras
%Normaliza dados (pre-
processamento)
DB_X1_Norm = Normaliza( 1, -1,
DB_X1 );
DB_X2_Norm = Normaliza( 1, -1,
DB_X2 );
DB_X3_Norm = Normaliza( 1, -1,
DB_X3 );

x = [];
%monta matriz de entradas
for k=1: length(DB_X1_Norm)
     y = Perceptron_Executa(pesos,
[-1 DB_X1_Norm(k) DB_X2_Norm(k)
DB_X3_Norm(k)]' );
end;
```