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

Alunos: Luisa Helena Bartocci Liboni
Rodrigo de Toledo Caropreso

Data de Entrega: 18/06/2012

Redes Neurais Artificiais

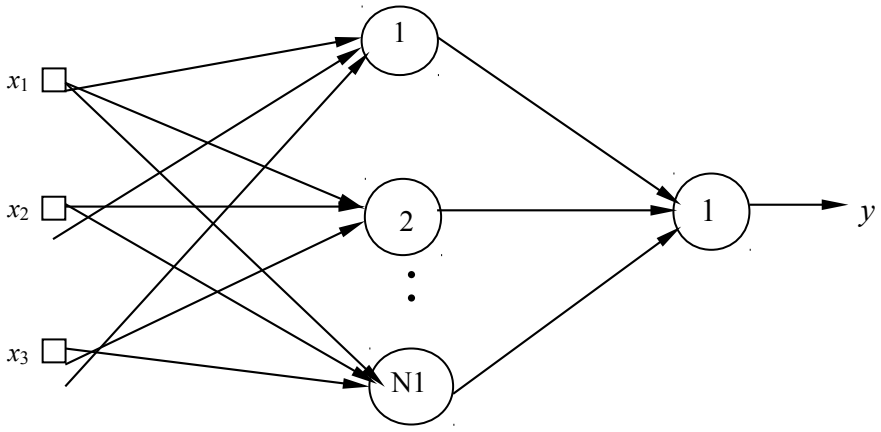
(Prof. Ivan Nunes da Silva)

EPC-10

A quantidade de gasolina $\{y\}$ a ser injetada por um sistema de injeção eletrônica de combustível para veículos automotores pode ser computada em tempo-real em função de três grandezas $\{x_1 , x_2 , x_3\}$. Devido à complexidade inerente do processo, configurado como um sistema não-linear, pretende-se utilizar uma rede neural artificial para o mapeamento entre as entradas e a saída do processo.

Sabe-se que para efetuar o respectivo mapeamento (problema de aproximação funcional), duas potenciais arquiteturas podem ser aplicadas, a saber, o perceptron multicamadas ou a RBF. Dado que a equipe de engenheiros e cientistas já realizou o mapeamento do problema por meio do perceptron multicamadas, o objetivo agora é treinar uma RBF a fim de que os resultados fornecidos por ambas as arquiteturas possam ser contrastados.

Assim, efetue o treinamento de uma RBF com o objetivo de computar a quantidade de gasolina $\{y\}$ a ser injetada pelo sistema de injeção eletrônica em função das variáveis $\{x_1 , x_2 , x_3\}$. A topologia da rede RBF está ilustrada na figura abaixo.



As topologias candidatas de RBF para serem aplicadas no mapeamento do problema acima são especificadas como se segue:

Rede 1 → RBF com N1 = 05
Rede 2 → RBF com N1 = 10
Rede 3 → RBF com N1 = 15

Utilizando os dados de treinamento apresentados no Anexo, execute o treinamento das redes RBF conforme as topologias definidas acima. Para tanto, faça as seguintes atividades:

1. Execute 3 treinamentos para cada topologia de rede RBF definida anteriormente, inicializando a matriz de pesos da camada de saída com valores aleatórios entre 0 e 1. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos das matrizes de pesos iniciais não sejam os mesmos. Utilize uma taxa de aprendizado $\eta = 0.01$ e precisão $\varepsilon = 10^{-7}$.

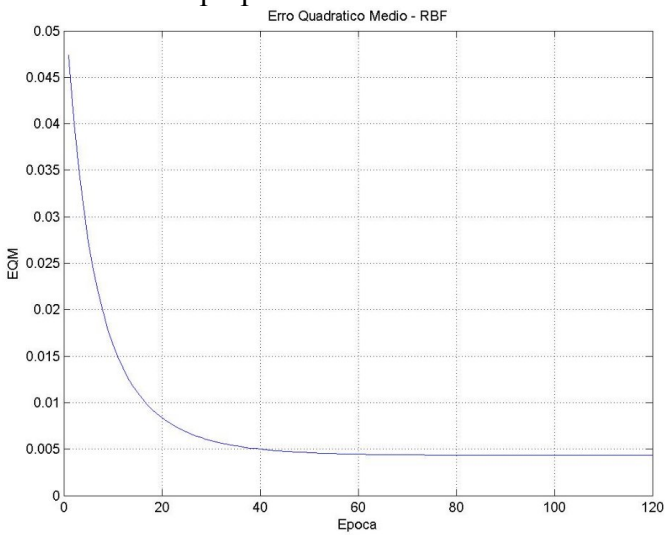
2. Registre os resultados finais desses 3 treinamentos, para cada uma das três topologias de rede, na tabela a seguir:

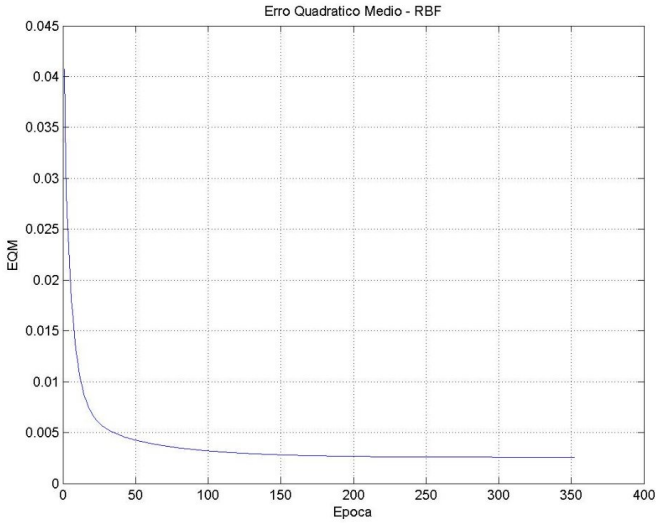
Treinamento	Rede 1		Rede 2		Rede 3	
	EQM	Épocas	EQM	Épocas	EQM	Épocas
T1	0,004322	121	0,002568	353	0,001861	789
T2	0,004323	130	0,002568	375	0,001861	810
T3	0,004323	131	0,002568	383	0,001861	839

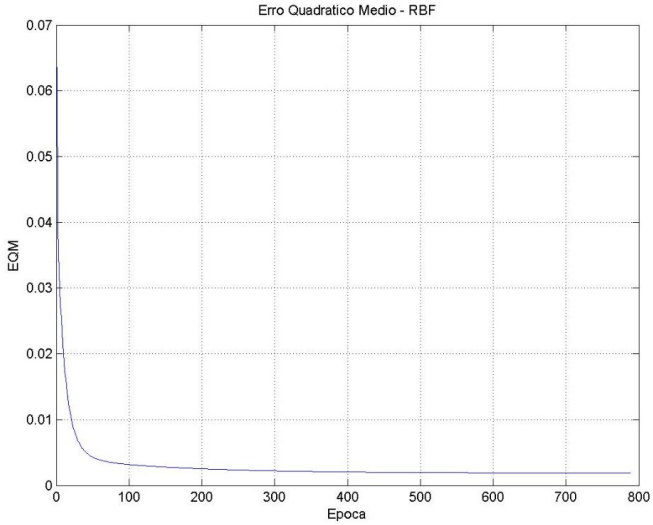
3. Para todos os treinamentos efetuados no item 2, faça a validação da rede em relação aos valores desejados apresentados na tabela abaixo. Forneça para cada treinamento o erro relativo médio (%) entre os valores desejados e os valores fornecidos pela rede em relação a todos os padrões de teste. Obtenha também a respectiva variância (%).

					Rede 1			Rede 2			Rede 3		
Amostra	x_1	x_2	x_3	d	y (T1)	y (T2)	y (T3)	y (T1)	y (T2)	y (T3)	y (T1)	y (T2)	y (T3)
01	0.5102	0.7464	0.0860	0.5965	0.6080	0.6079	0.6079	0.5820	0.5820	0.5820	0.5856	0.5856	0.5856
02	0.8401	0.4490	0.2719	0.6790	0.7363	0.7362	0.7362	0.6718	0.6719	0.6719	0.6538	0.6538	0.6538
03	0.1283	0.1882	0.7253	0.4662	0.4361	0.4360	0.4360	0.4804	0.4804	0.4804	0.5242	0.5242	0.5242
04	0.2299	0.1524	0.7353	0.5012	0.4430	0.4428	0.4429	0.5026	0.5026	0.5026	0.5426	0.5426	0.5426
05	0.3209	0.6229	0.5233	0.6810	0.7027	0.7026	0.7026	0.6704	0.6704	0.6704	0.6385	0.6385	0.6385
06	0.8203	0.0682	0.4260	0.5643	0.5961	0.5963	0.5962	0.5447	0.5447	0.5448	0.5664	0.5664	0.5664
07	0.3471	0.8889	0.1564	0.5875	0.5434	0.5434	0.5434	0.5847	0.5847	0.5847	0.6101	0.6101	0.6101
08	0.5762	0.8292	0.4116	0.7853	0.8289	0.8287	0.8288	0.7909	0.7909	0.7909	0.7456	0.7456	0.7456
09	0.9053	0.6245	0.5264	0.8506	0.8246	0.8245	0.8246	0.8792	0.8792	0.8792	0.8349	0.8349	0.8349
10	0.8149	0.0396	0.6227	0.6165	0.5946	0.5948	0.5948	0.6096	0.6096	0.6096	0.6929	0.6929	0.6929
11	0.1016	0.6382	0.3173	0.4957	0.4638	0.4638	0.4638	0.5066	0.5066	0.5066	0.5197	0.5197	0.5197
12	0.9108	0.2139	0.4641	0.6625	0.6504	0.6505	0.6505	0.6447	0.6447	0.6447	0.6092	0.6092	0.6092
13	0.2245	0.0971	0.6136	0.4402	0.3715	0.3713	0.3714	0.4421	0.4421	0.4421	0.4633	0.4633	0.4633
14	0.6423	0.3229	0.8567	0.7663	0.7149	0.7150	0.7149	0.7370	0.7370	0.7370	0.7080	0.7080	0.7080
15	0.5252	0.6529	0.5729	0.7893	0.8832	0.8831	0.8832	0.7726	0.7726	0.7726	0.7194	0.7194	0.7194
Erro Relativo Médio (%)					4.02	4.03	4.03	1.25	1.25	1.25	3.75	3.75	3.75
Variância (%)					0.0514	0.0515	0.0515	0.0078	0.0078	0.0078	0.0493	0.0493	0.0493

4. Para cada uma das topologias apresentadas na tabela acima, considerando ainda o melhor treinamento {T1, T2 ou T3} realizado em cada uma delas, trace o gráfico dos valores de erro quadrático médio (EQM) em função de cada época de treinamento. Imprima os três gráficos numa mesma folha de modo não superpostos.







5. Baseado nas análises dos itens acima, indique qual das topologias candidatas {Rede 1, Rede 2 ou Rede 3} e com que qual configuração final de treinamento {T1, T2 ou T3} seria a mais adequada para este problema.

Baseando-se nos resultados obtidos nos itens anteriores, a Rede 2 seria a configuração mais adequada para o problema em questão em razão de ter produzido o menor valor de Erro Relativo Médio e de Variância se comparado aos demais treinamentos de topologias analisadas.
Além disso, o treinamento T1 convergiu com menos épocas do que os demais treinamentos para a mesma topologia, então ele pode ser definido como o mais indicado neste caso.

Código fonte

```
clear;
clc;

id_treino = input('Identificador do
Treinamento(1-3): ');
id_rede   = input('Identificador da
Rede(1-3): ');

N1 = 5; %default

if( id_rede == 1 )
    N1 = 5;
end;

if( id_rede == 2 )
    N1 = 10;
end;

if( id_rede == 3 )
    N1 = 15;
end;

%Carrega os dados
Carrega_Tabela_Treino;

%Monta vetores de amostras
eta = 0.01; %coeficiente de treinamento
epson = 1e-07; % margem do erro

N_Entradas = 3; %entradas do RBF
n_camadas = 2;
size_Camadas = [N1 1];
N_Saidas = size_Camadas(2);
N_Camada_Oculta = size_Camadas(1);
N_Amostras = length(DB_X1);

%PRIMEIRO ESTAGIO DE TREINAMENTO - INICIO
X = [DB_X1 DB_X2 DB_X3];
[W_1 sigma] = RBF_Estagio1_Treino(X,
N_Entradas, N_Camada_Oculta);

disp 'Centroides'
W_1

disp 'Variancias'
sigma'

% Plota_Cluster( DB_X1, DB_X2, DB_X3,
DB_D, W_1, sigma );

% [cc, ss] = Caminhos(X, N_Camada_Oculta);
%
% disp 'Centroides'
% cc
% disp 'Variancias'
% ss
%
% W_1 = cc;
% sigma = ss;

pause
%PRIMEIRO ESTAGIO DE TREINAMENTO - FIM

%SAIDAS DA CAMADA NEURAL INTERMEDIARIA -
INICIO
[ Y_1 ] = RBF_Estagio1_Operacao( X, W_1,
sigma );
%SAIDAS DA CAMADA NEURAL INTERMEDIARIA -
FIM

%SEGUNDO ESTAGIO DE TREINAMENTO - INICIO
%monta matriz de entradas
x = [];
x = [ (-1)*ones(N_Amostras, 1) Y_1 ]';

d =  [DB_D]';
max_epocas = 20000;

%Treinamento
tic
[W_2, eqm, epoca ] =
RBF_Estagio2_Treino( eta, epson, x, d,
max_epocas, 1, N_Saidas );
t = toc
%SEGUNDO ESTAGIO DE TREINAMENTO - FIM

disp 'Pesos da Rede Treinada'
W_2
pause

%Grafico do EQM
plot( 1: length(eqm), eqm );
grid;
title( 'Erro Quadratico Medio - RBF');
xlabel( 'Epoca' );
ylabel( 'EQM' );
```

```
disp(sprintf('Epocas: %d, EQM: %f', epoca,
eqm(length(eqm))));

pause

%OPERACAO - BEGIN
%Carrega os dados
Carrega_Tabela_Operacao;

N_Amostras = length(DB_X1);

%Estagio 1
X = [DB_X1 DB_X2 DB_X3];
[ Y_1 ] = RBF_Estagio1_Operacao( X, W_1,
sigma );

Y_1
disp 'Pronto'

%Estagio 2
%monta matriz de entradas
x = [];
x = [ (-1)*ones(N_Amostras, 1) Y_1 ];

d =  [DB_D]';

y  = []; %saida real
yp = []; %saida pós processada
erro = []; %matriz de erro (d-y)

%Executa Rede
total_acertos = 0;
for k=1: N_Amostras

    y(:, k) = RBF_Estagio2_Operacao( W_2,
x(k, :)' );

    e(k) = y(:, k) - d(:, k);
end;


disp 'Saidas Reais do RBF';
y'

%Pos processamento
disp 'Saidas desejadas do RBF';
d'

disp 'Erro';
e'

disp 'Media e Variancia'
mean(e)
var(e)

%Salva dados
saveas( gcf, sprintf('Rede_%d_T_%d.jpg',
id_rede, id_treino) );

filename = sprintf('Dados_Rede_%d_T_
%d.txt', id_rede, id_treino);

fid = fopen(filename,'wt');
fprintf( fid,'EQM: %f  Epocas: %d\n' ,
eqm(length(eqm)), epoca );
fprintf( fid,'Saidas da Rede:\n' );
fprintf( fid,'%f\n' , y' );
fprintf( fid,'Media: %f  Variancia:
%f\n' , mean(e), var(e) );
fprintf( fid,'Media: %f %%  Variancia: %f
%%\n' , mean(e)*100, var(e)*100 );
fclose(fid);


function [ Y_1 ] =
RBF_Estagio1_Operacao( X, W_1, sigma )
%RBF_Estagio1_Operacao Executa o primeiro
estagio da RBF
%N_Amostras      -> quantidade de amostras
de entradas
%N_Camada_OCulta -> quantidade de funcoes
da camada intermediaria
%X               -> vetor de entradas
%W_1             -> matriz de pesos
(sinapses - cada linha = 1 neuron, cada
%coluna = 1 sinapse), basicamente as
coordenadas do centroide de cada
%funcao radial
%sigma           -> vetor de variancias
(cada linha = 1 neuron)

N_Amostras = size(X, 1); %qdte de linhas
= amostras
N_Entradas = size(X, 2); %qdte de colunas
= entradas da rede
N_Camada_Oculta = size(W_1, 1); %qdte de
linhas  = neurons

for k=1:N_Amostras
    for j=1:N_Camada_Oculta
        soma = 0;
        for i=1:N_Entradas
            soma = soma + ( X( k, i ) -
W_1( j, i ) )^2;
            Y_1(k, j) = exp(-soma/
(2*sigma(j)^2));
        end;
    end;
end;

function [W_1, var] =
RBF_Estagio1_Treino(X, N_Entradas,
N_Camada_Oculta)
%RBF_Estagio1 Executa o Primeiro Estagio
da RBF
%Calcula os centroides (pesos) e
variancias

%Vetor de inicialização do k-means
(alocaçao do "chute inicial de cada
%centroide")
start = [];
for i=1:N_Camada_Oculta
    start = [start; X(i,:)];
end;

% start = [ X(1,:); X(3,:) ];
% start
% pause

%Etapa1: Clusterização por K-Means
[idx, c] = kmeans( X, N_Camada_Oculta,
'Start', start );
```

```
% [idx, c] = kmeans( X, N_Camada_Oculta );

for j=1:size(c,1) %'j' -> cada linha
corresponde a um centroide
    soma = 0;
    for k=1:length(idx)
        if( idx(k) == j ) %verifica se
aquela amostra pertence ao cluster 'j'
            for i=1:N_Entradas
                soma = soma + (X(k, i) -
c(j, i))^2;
            end;
        end;
    end;
    sigma(j) = sqrt(soma /
length(find(idx==j))); %verifica quantos
elementos em idx pertencem ao cluster 'j'
end;

W_1 = c;
var = sigma;

function y = RBF_Estagio2_Operacao( W_2, X
)
%RBF_Estagio2_Operacao
%   X     -> matriz com entradas
%   y     -> vetor com saidas produzidas
pela rede
%   W_2   -> matriz de pesos da rede
treinada

I_2 = W_2 * X;
y = I_2; %usando função linear na saida

function [W_2, eqm, epoca ] =
RBF_Estagio2_Treino( eta, epson, entradas,
saidas, max_epocas, n_camadas,
size_Camadas )
%RBF_Estagio2 Treinamento de MLP
%   eta          -> coeficiente de treino
%   epson        -> margem de erro
%   entradas     -> matriz com entradas
%   saidas       -> vetor com saidas
desejadas
%   max_epocas   -> limite de epocas de
treinamento
%   n_camadas    -> numero de camadas
neurais da rede MLP
%   size_camadas -> vetor-linha com a
quantidade de neuronios em cada
%   camada

N_Entradas = size(entradas, 1);
N_Amostras = size(entradas, 2);

W_2 = rand(size_Camadas, N_Entradas);
%Matriz de pesos da camada 2 - 1 neuron =
1 linha; cada coluna é uma sinapse e tem
que incluir o bias da camada anterior (3
colunas)

disp('Inicialização da Rede MLP - Pesos
(Pressione uma tecla para continuar)');

%inicio do treinamento
epoca = 1;
eqm(epoca) = 1 + epson;
stop = 0;

%TREINAMENTO MLP 1 camada - BEGIN
difEQM = 1;
EQM_Atual = EQM( entradas, saidas, W_2 );
eqm(epoca) = EQM_Atual;

while (epson < difEQM && epoca <
max_epocas)
    EQM_Anterior = EQM_Atual;

    %1a Camada: Entrada -> Neurons da
camada oculta
    for k=1:N_Amostras
        %FORWARD -INICIO
        X = entradas( :, k );
        d = saidas  ( :, k );

        I_2 = W_2 * X;
        Y_2 = I_2; %usando função linear
na saida
        %FORWARD - FIM

        %BACKWARD - INICIO
        %Camada 2
        delta_2 =
GradienteLocalDeSaida( 1, d, Y_2 );

        %Atualiza pesos da camada de saida
        sizeW = size(W_2);
        for j=1:sizeW(1) %cada linha é um
neuron
            for i=1:sizeW(2) %cada coluna
é uma sinapse
                W_2(j,i) = W_2(j,i) + eta
* delta_2(j) * X(i);
            end;
        end;
        %BACKWARD - FIM
    end;

    %Calcula o Erro
    EQM_Atual = EQM( entradas, saidas, W_2
);

    eqm(epoca) = EQM_Atual;

    difEQM = abs (EQM_Atual -
EQM_Anterior);

    epoca = epoca + 1;
end;

%TREINAMENTO MLP 1 camada - END

if( epoca < max_epocas )
    disp( sprintf( 'Rede treinada. Numero
de epocas: %d', epoca) );
else
    disp( sprintf( 'Limite de epocas
atingido (%d), rede nao treinada.', epoca)
);
end;
```