1. Neural0.m

```
%
%  BACKPROPAGATION CODE.
%
% DEFINING INPUT PATTERNS AND OUTPUTS
% Esta es una versión revisada de la original. Agosto 1999.
% Gerardo Colmenares, Ph.D.
clc;

%
%  Main Menu
%
option=menu ('REDES NEURONALES MEDIANTE BACKPROPAGATION. Seleccione una
opción:',...
             ' Preparación de los datos',...
             ' Escalamiento de los datos',...
             ' Entrenamiento de la red neuronal',...
             ' Prueba del modelo de la red neuronal',...
             ' Mantenimiento de archivos',...
             ' Ninguna opción [Default]');
disp('')
%
if  option ==1
neural0a;
elseif  option ==2
neural0b;
elseif option ==3
neural3;
elseif option == 4
neural4;
elseif option ==5
neural5;
else
clc;
end
```

2. Neural0a.m

```
%
%  BACKPROPAGATION CODE.
%
% DEFINING INPUT PATTERNS AND OUTPUTS
%

clc;

%
%  Main Menu
%
option=menu ('La preparación de los datos es ',...
             ' Para la construcción del modelo',...
             ' Para la prueba del modelo',...
             ' Ninguna opción [Default]');
disp('');
```

```
%
if  option ==1
  neural1a;
elseif option ==2
  neural1b;
else
  neural0;
  clc;
end
```

3. Neural0b.m

```
%
%  BACKPROPAGATION CODE.
%
% DEFINING INPUT PATTERNS AND OUTPUTS
%

clc;

%
%  Main Menu
%
option=menu ('El escalamiento es ',...
             ' Para los datos de entrenamiento',...
             ' Para los datos de prueba del modelo',...
             ' Ninguna opción [Default]');
disp('');
%
if  option ==1
  neural2a;
elseif  option ==2
  neural2b;
else
   neural0;
  clc;
end
```

4. Neural1.m

```
%
%  BACKPROPAGATION CODE. (neural1)
%
% DEFINING INPUT PATTERNS AND OUTPUTS


%
% Create the files in a standar format
%
% Load the data matrix> in the next line include the name of file
% where is the matrix (the matrix name is P)
```

```
% PREPARATION OF MATRIX P
%======================
 clear all;
 echo off;
 final_row = 0;
 flag =0;
 flag3=0;

 clc
 flag4=menu('Los datos originales son:',...
                              'Un archivo en formato MAT',...
                              'Un archivo en formato ASCII',...
                              'Ninguna opción (Default)');
 disp('');

 clc;
 fprintf('\n\n PROCEDIMIENTO BACKPROPAGATION\n\n')
 fprintf(' Preparación de los datos\n\n')

 disp ('SELECCIONANDO UN ARCHIVO');
 if flag4 == 1
    flag3 = 1;
    IN  = input('Indique el nombre del archivo MAT: (entre apóstrofes) ');

       clc

    disp('Archivos MAT. Los datos deben estar en formato MAT');

    disp('')
    disp('Presione cualquier tecla para continuar');
    pause

    load(IN);
    disp('Desde la siguiente lista, seleccione el nombre del archivo');
    whos
    TEMPO = input('Indique el nombre seleccionado: ');

    clc

    disp('Cargando archivo MAT en TEMPO.');
    disp('Presione cualquier tecla al estar listo.');
    pause
    echo off

 elseif flag4==2
  flag=1;
  while(flag==1)
     disp('Indique unidad y nombre del archivo ASCII:');
     IN  = input('(entre apóstrofes. Ej:c:\mydata.dat): ');

    file = fopen(IN,'rw');
    if file == -1
      fprintf('\n\nError: Este archivo no existe. Trate de nuevo...')
      flag=1;
    else
      flag=0;
```

```
    end
  end

  fprintf ('\n\n Cargando el archivo...\n\n')

  data = fscanf(file,'%f');

  variables = input('Indique el total de variables (patrones): ');

  rows = length(data)/variables;

  data = reshape(data,rows,variables);
  TEMPO = data;
 end

 if flag3 == 1
  flag = input(' Otro archivo( 0 si / 1 no): ');
  fclose('all');
  flag3 =1;
 end
end


% DEFINING AND SAVING THE FINAL INPUT  PATTERNS AND OUTPUTS
% ========================================================
if flag3 == 1
  clc
  type_data=menu('Clase de datos a guardar:',...
                              'Ejemplos de salida para la red',...
                              'Ejemplos de entrada para la red');
  disp('')
  if type_data ==1
    Out=TEMPO;
    save Output.mat Out
    clear TEMPO;
  else
    Inp=TEMPO;
    save Input.mat Inp
    clear TEMPO;
 end
end
```

5. Neural1a.m

```
%
%  BACKPROPAGATION CODE. (neural1a)
%
% DEFINING INPUT PATTERNS AND OUTPUTS


%
% Create the training files in a standar format
%
% Load the data matrix> in the next line include the name of file
% where is the matrix (the matrix name is P)
```

```
% PREPARATION OF MATRIX P
%=======================
 clear all;
 echo off;
 final_row = 0;
 flag =0;
 flag3=0;

 clc
 flag4=menu('Los datos originales son:',...
                              'Un archivo en formato MAT',...
                              'Un archivo en formato ASCII',...
                              'Ninguna opción (Default)');
 disp('');

 clc;
 fprintf('\n\n PROCEDIMIENTO BACKPROPAGATION\n\n')
 fprintf(' Preparación de los datos\n\n')

 disp ('SELECCIONANDO UN ARCHIVO');
 if flag4 == 1
    flag3 = 1;
    IN  = input('Indique la ruta y archivo MAT: (entre apóstrofes) ');

       clc

    disp('Archivos MAT. Los datos deben estar en formato MAT');

    disp('')
    disp('Presione cualquier tecla para continuar');
    pause

    load(IN);
    disp('Desde la siguiente lista, seleccione el nombre del archivo');
    whos
    TEMPO = input('Indique el nombre seleccionado: ');

    clc

    disp('Cargando archivo MAT en TEMPO.');
    disp('Presione cualquier tecla al estar listo.');
    pause
    echo off

 elseif flag4==2
  flag=1;
  while(flag==1)
    disp('Indique la ruta y archivo ASCII:');
    IN  = input('(entre apóstrofes. Ej:c:\mydata.dat): ');

    file = fopen(IN,'rw');
    if file == -1
      fprintf('\n\nError: Este archivo no existe. Trate de nuevo...')
      flag=1;
```

```
    else
      flag=0;
    end
  end
  fprintf ('\n\n Cargando el archivo...\n\n')

  data = fscanf(file,'%f');

  variables = input('Indique el total de variables (patrones): ');

  rows = length(data)/variables;

  data = reshape(data,rows,variables);
  TEMPO = data;
 else
   neural0a;
 end

 if flag3 == 1
  flag = input(' Otro archivo( 0 si / 1 no): ');
  fclose('all');
  flag3 =1;
 end


% DEFINING AND SAVING THE FINAL INPUT  PATTERNS AND OUTPUTS
% ========================================================
if flag3 == 1
  clc
  type_data=menu('Clase de datos a guardar:',...
                              'Ejemplos de salida para la red',...
                              'Ejemplos de entrada para la red');
  disp('')
  if type_data ==1
     Out=TEMPO;
     figure;
     plot(Out);
     save Output.mat Out
    clear TEMPO;
  else
    Inp=TEMPO;
    save Input.mat Inp
    clear TEMPO;
 end
 neural0a;
end
```

6. Neural1b.m

```
%
%  BACKPROPAGATION CODE. (neural1b)
%
% DEFINING INPUT PATTERNS AND OUTPUTS
```

```
%
% Create the testing files in a standar format
%
% Load the data matrix> in the next line include the name of file
% where is the matrix (the matrix name is P)



% PREPARATION OF MATRIX P
%======================
 clear all;
 echo off;
 final_row = 0;
 flag =0;
 flag3=0;

 clc
 flag4=menu('Los datos originales son:',...
                              'Un archivo en formato MAT',...
                              'Un archivo en formato ASCII',...
                              'Ninguna opción (Default)');
 disp('');

 clc;
 fprintf('\n\n PROCEDIMIENTO BACKPROPAGATION\n\n')
 fprintf(' Preparación de los datos\n\n')

 disp ('SELECCIONANDO UN ARCHIVO');
 if flag4 == 1
    flag3 = 1;
    IN  = input('Indique la ruta y archivo MAT: (entre apóstrofes) ');

       clc

    disp('Archivos MAT. Los datos deben estar en formato MAT');

    disp('')
    disp('Presione cualquier tecla para continuar');
    pause

    load(IN);
    disp('Desde la siguiente lista, seleccione el nombre del archivo');
    whos
    TEMPO = input('Indique el nombre seleccionado: ');

    clc

    disp('Cargando archivo MAT en TEMPO.');
    disp('Presione cualquier tecla al estar listo.');
    pause
    echo off

 elseif flag4==2
  flag=1;
  while(flag==1)
```

```
     disp('Indique la ruta y archivo ASCII:');
     IN  = input('(entre apóstrofes. Ej:c:\mydata.dat): ');

    file = fopen(IN,'rw');
    if file == -1
      fprintf('\n\nError: Este archivo no existe. Trate de nuevo...')
      flag=1;
    else
      flag=0;
    end
  end

  fprintf ('\n\n Cargando el archivo...\n\n')

  data = fscanf(file,'%f');

  variables = input('Indique el total de variables (patrones): ');

  rows = length(data)/variables;

  data = reshape(data,rows,variables);
  TEMPO = data;
 end

 if flag3 == 1
  flag = input(' Otro archivo( 0 si / 1 no): ');
  fclose('all');
  flag3 =1;
 end


% DEFINING AND SAVING THE FINAL INPUT  PATTERNS AND OUTPUTS
% ========================================================
if flag3 == 1
  clc
  type_data=menu('Clase de datos a guardar:',...
                              'Ejemplos de salida para la red',...
                              'Ejemplos de entrada para la red');
  disp('')
  if type_data ==1
     Out=TEMPO;
     figure;
     plot(Out)
     save TestOut.mat Out
     clear TEMPO;
  else
     Inp=TEMPO;
     save TestInp.mat Inp
     clear TEMPO;
  end
  neural0a;
end
```

8. Neural2.m

```
%
%  BACKPROPAGATION CODE. (neural2)
%
%  SCALING OUTPUT AND/OR INPUT EXAMPLES

%

% DEFINING  THE FINAL INPUT  PATTERNS AND OUTPUTS
% ==============================================

clc
fprintf('Escalamiento de los datos\n\n')

type_data=menu('Clase de datos a ser escalados:',...
                              'Ejemplos de salida de la red',...
                              'Ejemplos de entrada de la red',...
                              'Ninguna opción [Default]');
disp('')
if type_data ==1
  load Output;
  TEMPO= Out;
elseif type_data ==2
  load Input;
  TEMPO=Inp;
else
return
end


flag1 = input('Quiere hacer escalamiento( 0 si / 1 no): ');

if flag1==0
  fprintf ('\n\nRango de escalamiento...');
  tmin=input('\n Valor mínimo en la escala: ');
  tmax=input(' Valor máximo en la escala: ' );

%  P3 = scaling(TEMPO,tmin,tmax);

  fprintf('\n\nEscalando....\n\n');
  A=TEMPO;
  [M,N] = size(A);
  V=[min(A);max(A)];
  units=ones(M,N);
  P1=A-units(1:M,1:1)*V(1:1,1:N);
  for l=1:M;
   for k=1:N;
    if V(2,k)-V(1,k) ~= 0;
      P1(l,k)=(P1(l,k))/(V(2,k)-V(1,k));
      P1(l,k) = tmin + P1(l,k)*(tmax-tmin);
    end;
   end;
  end;
  clear A;
  clear TEMPO;
  TEMPO = P1';
```

```
  clear P1;
  if type_data ==1
    Out=TEMPO;
    save Output.mat Out
  else
    Inp=TEMPO;
    save Input.mat Inp
  end
end
```

9. Neural2a.m

```
%
%  BACKPROPAGATION CODE. (neural2)
%
%  SCALING OUTPUT AND/OR INPUT EXAMPLES

%

% DEFINING  THE FINAL INPUT  PATTERNS AND OUTPUTS
% ==============================================

clc
fprintf('Escalamiento de los datos\n\n')

type_data=menu('Clase de datos a ser escalados:',...
                              'Ejemplos de salida de la red',...
                              'Ejemplos de entrada de la red',...
                              'Ninguna opción [Default]');
disp('')
if type_data ==1
  load Output;
  TEMPO= Out;
elseif type_data ==2
  load Input;
  TEMPO=Inp;
else
return
end


flag1 = input('Quiere hacer escalamiento( 0 si / 1 no): ');

if flag1==0
  fprintf ('\n\nRango de escalamiento...');
  tmin=input('\n Valor mínimo en la escala: ');
  tmax=input(' Valor máximo en la escala: ' );

%  P3 = scaling(TEMPO,tmin,tmax);

  fprintf('\n\nEscalando....\n\n');
  A=TEMPO;
  [M,N] = size(A);
  V=[min(A);max(A)];
  units=ones(M,N);
```

```
  P1=A-units(1:M,1:1)*V(1:1,1:N);
  for l=1:M;
   for k=1:N;
    if V(2,k)-V(1,k) ~= 0;
      P1(l,k)=(P1(l,k))/(V(2,k)-V(1,k));
      P1(l,k) = tmin + P1(l,k)*(tmax-tmin);
    end;
   end;
  end;
  clear A;
  clear TEMPO;
  TEMPO = P1;
  clear P1;
  if type_data ==1
    Out=TEMPO;
    save Output.mat Out
  else
    Inp=TEMPO;
    save Input.mat Inp
 end
 neural0b;
end
```

10. Neural2b.m

```
%
%  BACKPROPAGATION CODE. (neural2)
%
%  SCALING OUTPUT AND/OR INPUT EXAMPLES

%

% DEFINING  THE FINAL INPUT  PATTERNS AND OUTPUTS
% ===============================================

clc
fprintf('Escalamiento de los datos\n\n')

type_data=menu('Clase de datos a ser escalados:',...
                              'Ejemplos de salida de la red',...
                              'Ejemplos de entrada de la red',...
                              'Ninguna opción [Default]');
disp('')
if type_data ==1
  load TestOut;
  TEMPO= Out;
elseif type_data ==2
  load TestInp;
  TEMPO=Inp;
else
return
end
```

```
flag1 = input('Quiere hacer escalamiento( 0 si / 1 no): ');

if flag1==0
  fprintf ('\n\nRango de escalamiento...');
  tmin=input('\n Valor mínimo en la escala: ');
  tmax=input(' Valor máximo en la escala: ' );

%  P3 = scaling(TEMPO,tmin,tmax);

  fprintf('\n\nEscalando....\n\n');
  A=TEMPO;
  [M,N] = size(A);
  V=[min(A);max(A)];
  units=ones(M,N);
  P1=A-units(1:M,1:1)*V(1:1,1:N);
  for l=1:M;
   for k=1:N;
    if V(2,k)-V(1,k) ~= 0;
      P1(l,k)=(P1(l,k))/(V(2,k)-V(1,k));
      P1(l,k) = tmin + P1(l,k)*(tmax-tmin);
    end;
   end;
  end;
  clear A;
  clear TEMPO;
  TEMPO = P1;
  clear P1;
  if type_data ==1
    Out=TEMPO;
    save TestOut.mat Out
  else
    Inp=TEMPO;
    save TestInp.mat Inp
 end
 neural0b;
end
```

11. Neural3.m

```
%
%  BACKPROPAGATION CODE.
%
% Matrix for Backpropagation

clc;
fprintf('\n\n                                    ENTRENAMIENTO mediante
Backpropagation\n\n')
fprintf('\n\n                                    Modelo de red neuronal basado en
tres capas\n\n')

load Output.mat;
load Input.mat;
```

```
T1='logsig';
T2='tansig';

% DEFINING FINAL INPUT  PATTERNS AND OUTPUTS
% ======================================

[M,N]=size(Inp');
P=Inp';
T=Out';


% INPUT PATTERN AND OUTPUT
% ------------------------------------------------------
% M is the number of components in each input pattern

number_inputs=M;
number_patterns=N;


%
% INITIALIZE NETWORK ARCHITECTURE
% ==============================
% Set input vector size R, layer sizes S1 & S2, batch size Q.

fprintf('\n Inicialización. Espere.......')
[R,C]=size(P);
[R1,C1]=size(T);

% S1: number of hidden neurons (This value can be changed)

S1=fix((R+R1)/2);

S2=R1;
IN  =menu('Seleccione el proceso:',...
                'Iniciando el proceso de la red',...
                'Reprocesando la red');

disp('')
fprintf('\n Cargando los pesos.....\n');
if IN == 1
% RANDS Symmetric random generator.
%  W1=rands(S1,R);
%  B1=rands(S1,1);
%  W2=rands(S2,S1);
%  B2=rands(S2,1);
% NWLOG Nguyen-Widrow random generator for LOGSIG neurons.
  [W1,B1]=nwlog(S1,R);
  [W2,B2]=nwlog(S2,S1);
else
  load whgts.mat
% The next line is used when you want to use old weights
  W1=NW1; B1=NB1;W2=NW2;B2=NB2;
end
% ------------------------------------------------------

% TRAIN THE NETWORK
% =================
```

```
% TRAINING PARAMETERS

% Parameters:
% desired error: error_goal
% maximumm number of epochs: max_epochs
% learning rate: lr
%
% This parameters can be changed

  disp_freq =10;
  max_epochs=200;
  error_goal=0.003;
  lr = 0.01;
  lr_inc = 1.05;
  lr_dec = 0.7;
  mom_const = 0.95;
  err_ratio = 1.04;
  F1 ='logsig';
  F2 = 'logsig';

% Displaying the parameters by default

  fprintf('\n PARAMETROS PRESELECCIONADOS:\n')
  fprintf('    Frecuencia de resultados: %d \n    Número máximo de pasos
(epochs): %d \n',disp_freq,max_epochs)
  fprintf('    Error permitido: %f \n    Tasa de Aprendizaje: %f
\n',error_goal,lr)
  fprintf('    Incremento del aprendizaje: %f \n    Momento:
%f\n',lr_inc,mom_const)
  fprintf('    Función de activación para la capa oculta: %s \n',F1)
  fprintf('    Función de activación para la capa de salida: %s\n',F2)
  fprintf('    El número de nodos en la capa oculta sugerido es: %d\n',S1)

par = menu('Selección de los parámetros de entonación de la red',...
                'Fijar nuevos prámetros',...
                'Establecer los parámetros sugeridos');

disp('');

if par ==1

% Input the new parameters


  S1 = input(' Número de nodos ocultos ');
  if size(S1) == [0,0]
     S1 = fix((R+R1)/2);
  end

  disp_freq=input(' Frecuencia de la presentación de resultados: ');
  if size(disp_freq) == [0,0]
     disp_freq = 10;
  end


  max_epochs=input(' Máximo número de pasos(epochs): ');
```

```
  if size(max_epochs) == [0,0]
     max_epochs = 200;
  end


  error_goal = input(' Error permitido de convergencia: ');
  if size(error_goal) == [0,0]
     error_goal = 0.003;
  end


  lr = input(' Tasa de aprendizaje: ');
  if size(lr) == [0,0]
     lr = 0.01;
  end


  lr_inc = input(' Incremento de la tasa de aprendizaje: ');
  if size(lr_inc) == [0,0]
     lr_inc = 1.05;
  end

  mom_const= input(' Nuevo momento: ');
  if size(mom_const) == [0,0]
     mom_const = 0.95;
  end

  fprintf(' Typo de función de activación: tansig (-1,1), logsig (0,1)\n');

  F1 = input(' Función de activación para la capa oculta: ', 's');
  if F1 ~= T1 | F1 ~= T2
     F1='logsig';
  end

  F2 = input(' Función de activación para la capa de salida: ','s');
  if F2 ~= T1 | F2 ~= T2
     F2 = 'logsig';
  end

end


% NOTE: El resto del codigo de entrenamiento puede ser reemplazado
%       mediante la función:
%
% [W1,B1,W2,B2,epoch,TR] = trainbp(W10,B10,'tansig', ...
%        W20,B20,'purelin',P,T,TP);


% Learning phase

fprintf('\n\n ENTRENADO LA RED. Espere.......\n\n')
figure;
TP = [disp_freq max_epochs error_goal lr lr_inc lr_dec mom_const err_ratio];

[NW1,NB1,NW2,NB2,epoch,error] = ...
      trainbpx(W1,B1,F1,W2,B2,F2,P,T,TP);
```

```
Actual= logsig(W2*logsig(W1*P,B1),B2);


% PLOT ERROR CURVE
%=================
figure;
ploterr(error);
pause


fprintf('\n\n Convergence. Wait.......\n\n')

save whgts.mat NW1 NB1 NW2 NB2;
neural0;
```

1. Neural4.m

```
%
%  BACKPROPAGATION CODE. (neural4)
%
%  Use ouput.mat and input.mat matrix for Testing

fprintf('\n\n                          ENTRENAMIENTO mediante
Backpropagation\n\n')
fprintf('\n\n                          Prueba del modelo de red neuronal\n\n')

load TestOut;
load TestInp;



% DEFINING FINAL INPUT  PATTERNS AND OUTPUTS
% ======================================

[M,N]=size(Inp');



% INPUT PATTERN AND OUTPUT
% ------------------------------------------------------
% M is the number of components in each input pattern

number_inputs=M;
number_patterns=N;



% DEFINING FINAL INPUT  PATTERNS AND OUTPUTS
% ======================================

P=Inp';
T=Out';
```

```
clear Inp; clear Out;

% INPUT PATTERN AND OUTPUT
% ------------------------------------------------------

fprintf('\n\n Testing the network. Wait.......\n\n')
fprintf('    Desired     Actual    Squared \n')
fprintf('    Output      Output     Error \n')


%
% INITIALIZE NETWORK ARCHITECTURE
% ==============================
%



load whgts.mat

% The next line is used when you want to use old weights

W1=NW1; B1=NB1;W2=NW2;B2=NB2;

% TESTING THE NETWORK
% ==================


SSE=sumsqr(T-logsig(W2*logsig(W1*P,B1),B2));

Actual= logsig(W2*logsig(W1*P,B1),B2);

fprintf(' %10.5f %10.5f %10.5f \n',  T, Actual, SSE)


fprintf('\n\n End Testing......\n\n')
figure;
plot(T),hold,plot(Actual,'r');

fprintf('\n\n Saving Results (Target output, Actual output, Sum Squared
Error)');


save results.mat T Actual SSE;
```

12. Neural5.m

```
%
%  BACKPROPAGATION CODE. (neural5)
%
% SAVING OUPT AND INPUT EXAMPLES AS TRAINING OR TESTING DATASETS

%

% DEFINING  THE FINAL INPUT  PATTERNS AND OUTPUTS
% ==============================================
```

```
clc
fprintf(' Mantenimiento de Datos\n\n')
type_data1=menu('Tipo de datos a respaldar',...
                                'Datos de entrenamiento',...
                                'Datos de prueba',...
                                'Modelo de la red neural entrenada',...
                                'Ninguna opción [Default]');
disp('')

if type_data1 ==1
  type_data=menu('Datos de entrenamiento a respaldar',...
                                'Ejemplos de salida',...
                                'Ejemplos de entrada',...
                                'Ambos',...
                                'Ninguna opción [Default]');
elseif type_data1==2
  type_data=menu('Datos de prueba a respaldar',...
                                'Ejemplos de salida',...
                                'Ejemplos de entrada',...
                                'Ambos',...
                                'Ninguna opción [Default]');
elseif type_data1==3
   fprintf(' La red neural respaldada en net.mat contiene la matriz\n')
   fprintf(' de pesos sinápticos de las capas oculta y de salida\n')
   load whgts.mat
   % The next line is used when you want to use old weights
   save net.mat NW1 NB1 NW2 NB2;
   type_data = 4;
   neural0
end
disp('')
if type_data ==1
   if type_data1 ==1
       load Output;
       save Train.mat Out
       neural0
   elseif type_data1 ==2
       load TestOut;
       save Test.mat Out
       neural0
   end
elseif type_data ==2
   if type_data1 ==1
       load Input;
       save Train.mat Inp
       neural0
   elseif type_data1 ==2
       load TestInp;
       save Test.mat Out Inp
       neural0
   end
elseif type_data ==3
   if type_data1 ==1
       load Input;
       load Output;
       save Train.mat Out Inp
       neural0
```

```
   elseif type_data1 ==2
       load TestInp;
       load TestOut;
       save Test.mat Out Inp
       neural0
   end
else
   neural0;
end;
```