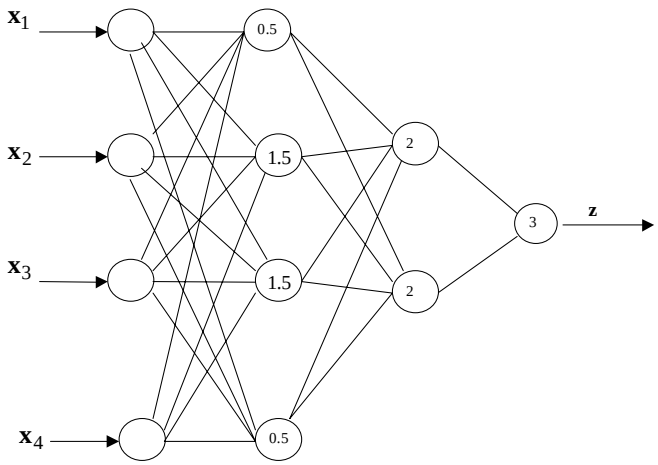
CAPÍTULO 2

TLU(s) MULTICAPAS

Se pueden implementar funciones en TLU con más de una capa.



No existen, en la actualidad, mecanismos de entrenamiento que permita conocer los pesos sinápticos de las redes TLU multicapas. Algunos mecanismos de aprendizaje, como el Adaline, el Procesador cuadrático, el Madaline, la máquina comité, etc, pretenden resolver la falta de captación no lineal de los algoritmos seminales de entrenamiento. Widrow y Hoff (1960), introducen el algoritmo Least Mean-Squared (LMS) para derivar los adaptadores lineales ADALINE y MADALINE en problemas de clasificación.

Esto por supuesto conduce a pensar:

1. Los TLU resuelven separabilidad en la clasificación de problemas típicamente lineales. Es decir son buenos clasificadores lineales.
2. Los TLU multicapas podrían resolver teóricamente cualquier problema pero no se dispone de procedimientos y mucho menos de algoritmos generales de entrenamiento para estos casos.

Backpropagation (Retropropagación) surge como un método alternativo para resolver la generalidad no alcanzada por los TLU.

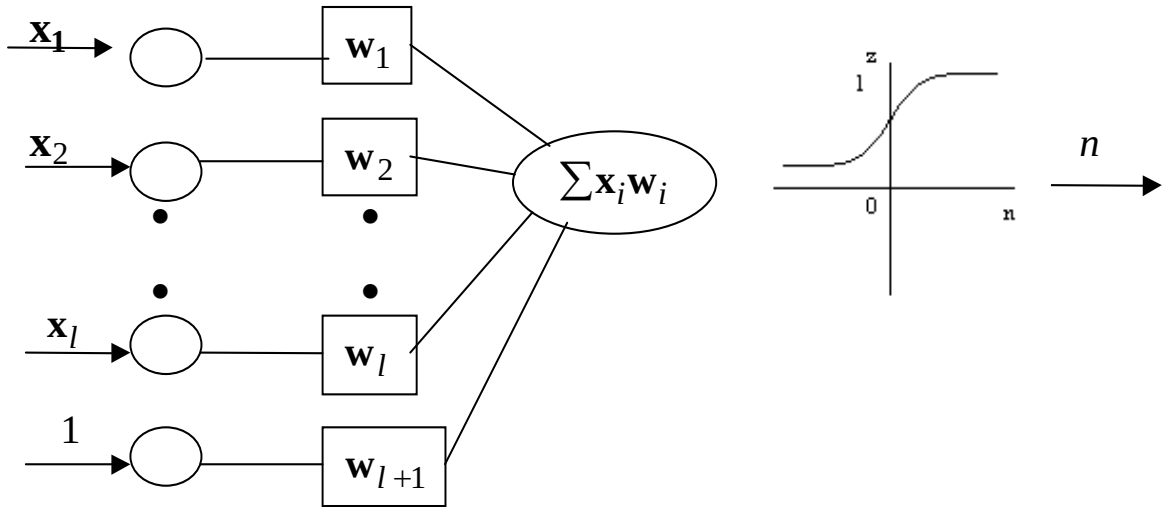
BACKPROPAGATION

Qué es Backpropagation o Retropropagación?

1. También conocido como retropropagación del error.
2. Método para calcular el gradiente del error

Recordemos lo que se mencionó al principio de este texto. (Werbos, 1974) (Parker, 1985) (Rumelhart, 1986). Se desarrolla el algoritmo Backpropagation como un mecanismo de aprendizaje para perceptrones multicapas. De alta popularidad para la solución de problemas de clasificación y pronóstico.

Es un método de entrenamiento general para redes multicapas, requiriendo diferenciabilidad en la capa de salida. Es decir la función debe ser diferenciable (sigmoide). En la siguiente figura podemos observar una neurona básica en backpropagation.

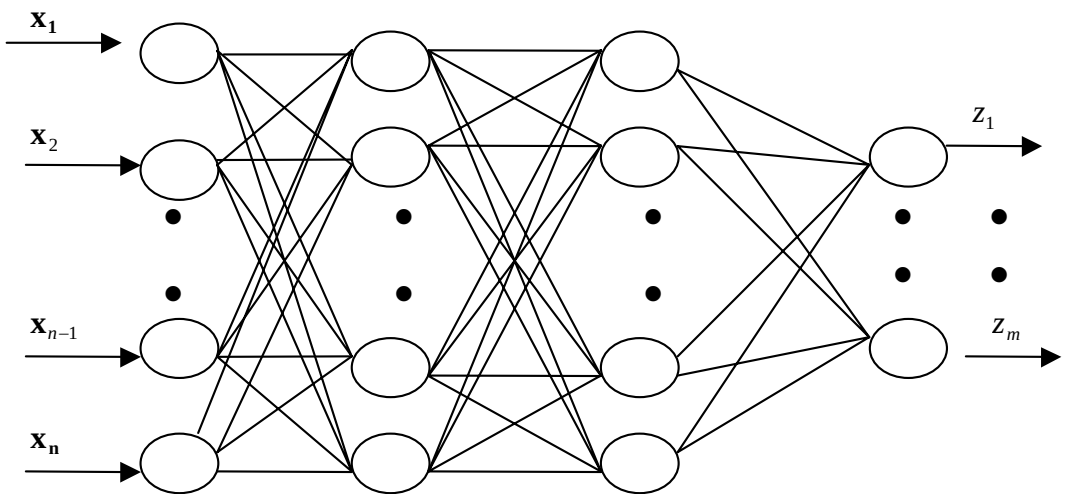


$$\mathbf{z} = \frac{1}{1 + e^{-n}}$$

si n >> 0 entonces z → 1
si n << 0 entonces z → 0

1. Las neuronas transforman una entrada no- restringida en una señal limitada Z
2. La función sigmoidal restringe el rango de Z entre 0 y 1.
3. Es diferenciable. Hay que tener presente que la no linealidad es una característica resaltante de este algoritmo.
4. Las entradas a la unidad de procesamiento son ejemplos de entrada o salidas de la capa previa
5. La función sigmoide obliga que el rango de valores de Z estén entre 0 y 1.

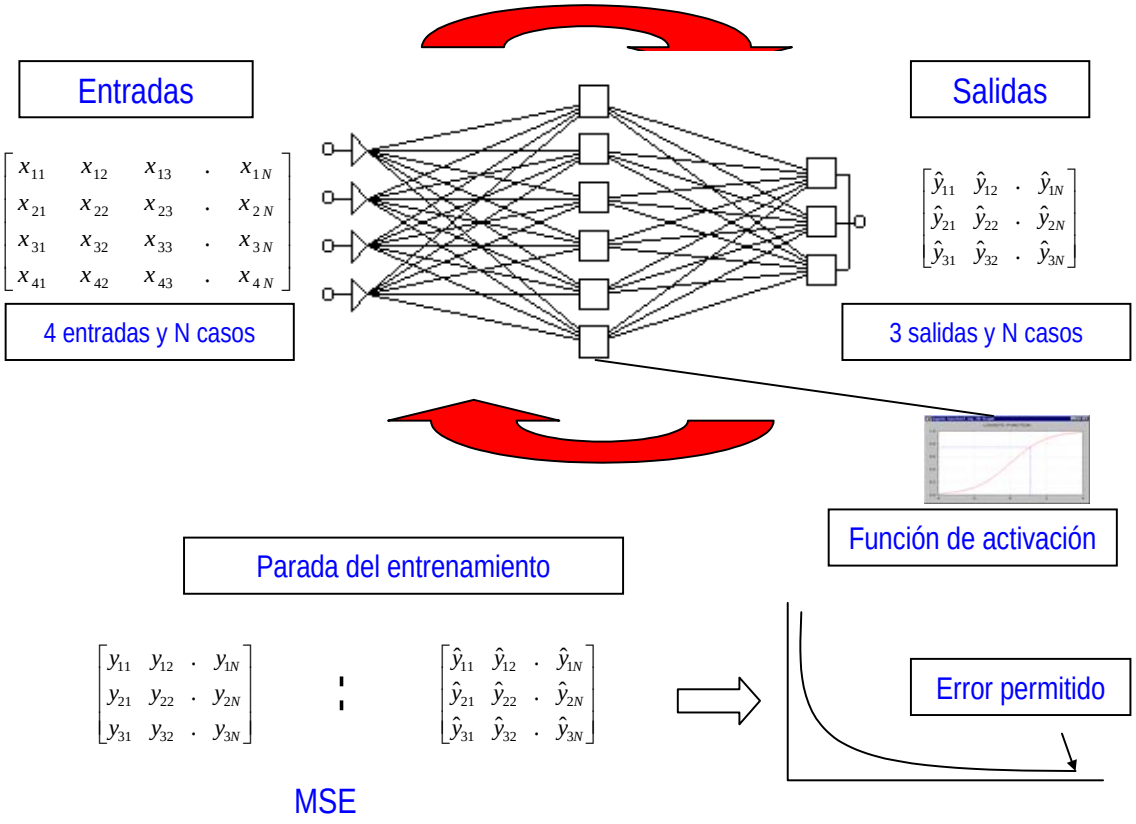
En una red cuya arquitectura incluye capas ocultas con varias neuronas (nodos) ocultas, podemos observar las siguientes características de diseño.



capa de entrada | capa oculta | capa oculta | capa de salida

- Está conectada hacia delante.
- Esta red en particular tiene la capa de entrada, dos capas ocultas y la capa de salida.

- La capa de entrada sirve solamente como un punto de distribución.
- Una neurona esta asociada con un conjunto de pesos que conecta a sus entradas. Los pesos en la capa uno terminan en las neuronas de la capa uno.



En backpropagation, el método general de entrenamiento se resume a cuatro pasos:

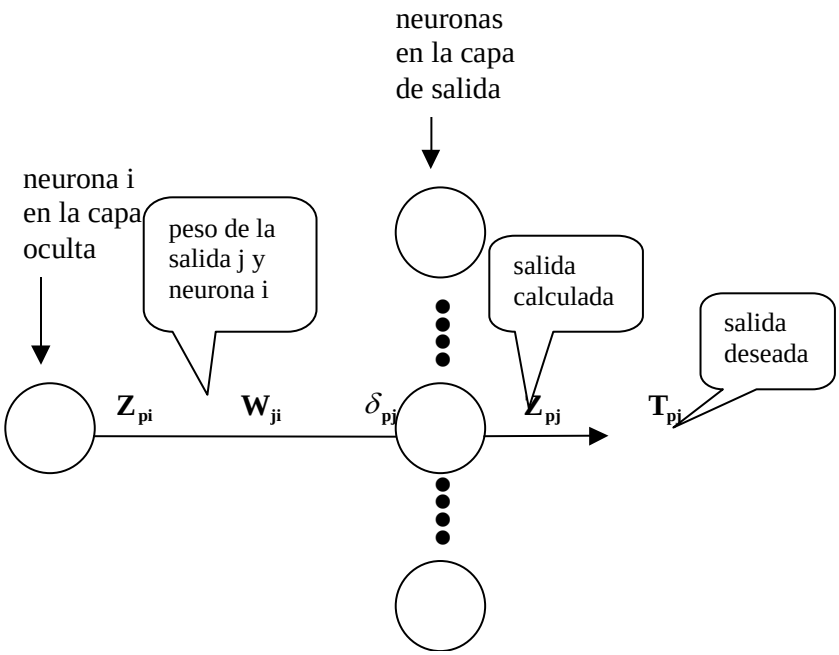
Pasos hacia delante:

1. Selecciona un vector de entrada desde el conjunto de entrenamiento.
2. Aplica esta entrada a la red y calcula la salida.

Pasos hacia atrás:

3. Calcular el error entre la salida calculada y la salida deseada de la entrada usada.
4. Ajustar los pesos para que el error cometido entre la salida calculada y la salida deseada sea disminuido.
5. Repetir los pasos 1 al 4 para todas las entradas del conjunto de entrenamiento, hasta que el error global sea aceptablemente bajo.

COMO AJUSTA LOS PESOS DE LA CAPA DE SALIDA

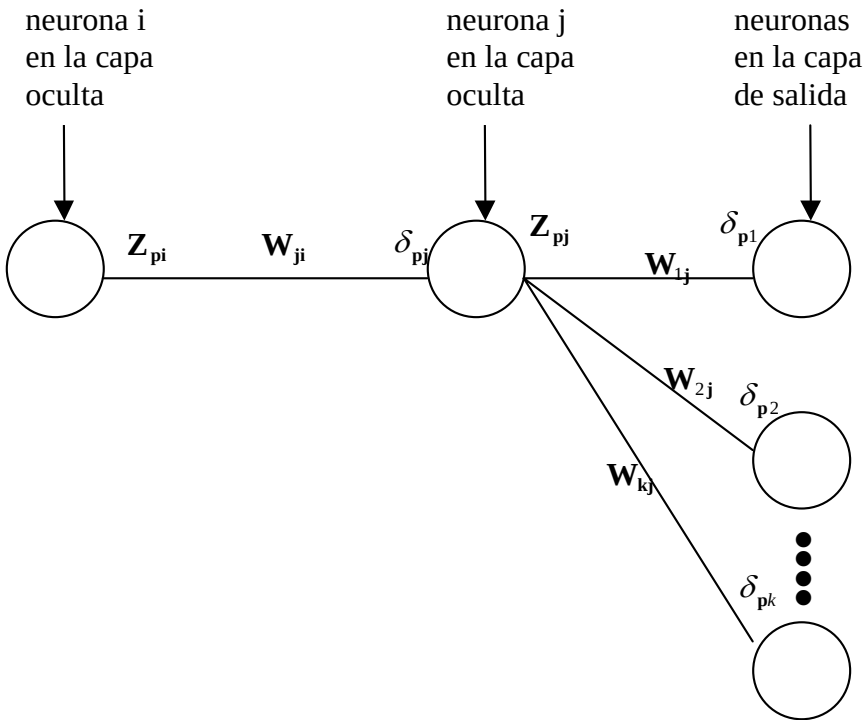


Como el valor de la salida deseado está disponible en la capa de salida, entonces los pesos son ajustados usando algún algoritmo tal como regla Delta y las constantes de aprendizaje.
Veamos. Para un patrón de entrada *p* determinado, si W_{ji} es el peso que se corresponde con la salida j y la neurona oculta i, entonces el valor de corrección entre la salida deseada y la calculada para ese patrón *p* expresada por δ_{pj} y la salida correspondiente a la neurona i dada por Z_{pi}, podría corregir el valor del peso W_{ji}. Es decir,

$\Delta_{\mathbf{p}} \mathbf{W}_{\mathbf{ji}} = \eta \delta_{\mathbf{pj}} \mathbf{Z}_{\mathbf{pi}}$, donde

$\delta_{\mathbf{pj}} = \mathbf{Z}_{\mathbf{pj}}(1 - \mathbf{z}_{\mathbf{pj}})(\mathbf{T}_{\mathbf{pj}} - \mathbf{Z}_{\mathbf{pj}})$

COMO AJUSTA LOS PESOS ENTRE CAPAS OCULTAS



Por otro lado, la actualización de los pesos entre las capas ocultas sería calculando el δ propagando hacia atrás desde la capa de salida.

$$\delta_{\mathbf{pj}} = \mathbf{Z_{pj}}(1 - \mathbf{Z_{pj}})\left(\sum_{\mathbf{k}} \delta_{\mathbf{pk}} \mathbf{w_{kj}}\right)$$

$$\Delta_{\mathbf{p}} \mathbf{w_{ji}} = \eta \delta_{\mathbf{pj}} \mathbf{Z_{pi}}$$

Este cálculo se repite en las capas ocultas de la red.

COMO SE DERIVA LAS FORMULAS

1. Backpropagation es un método gradiente descendente.

2. Minimiza el cuadrado de las diferencias entre las valores calculados y deseados de las salidas sumados sobre todas las unidades de salida y todo los pares, entradas y salidas, del conjunto de entrenamiento

3. Para un par *p* de entrenamiento (entrada y salidas) y *n* unidades de salida,

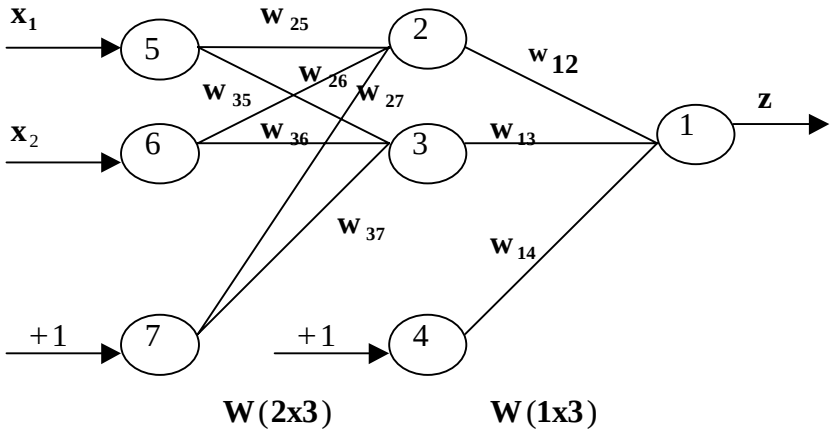
$$\mathbf{E_p} = \frac{1}{2}\sum_{\mathbf{j}=1}^{\mathbf{n}}\left(\mathbf{T_{pj}} - \mathbf{Z_{pj}}\right)^2$$

4. El error total sobre todo los patrones sería

$$\mathbf{E} = \sum_{\mathbf{p}} \mathbf{E_p}$$

UN EJEMPLO USANDO BACKPROPAGATION

Sea la red



Asuma una función sigmoide tal que $f'(n) = Z_{pi}(1-Z_{pi})$

Asuma $\eta = 1$ y todos los pesos inicialmente a 1.0.

El conjunto de entrenamiento es 0 0 → 1
0 1 → 0

Cómo sería un ciclo de aprendizaje?

Para el primer patrón de entrada 0 0 → 1

Pasos hacia delante:

$n2 = w_{25}\, x_1 + w_{26}\, x_2 + w_{27}\,(+1) = 1*0 + 1*0 + 1*1 = 1$

$n3 = w_{35}\, x_1 + w_{36}\, x_2 + w_{37}\,(+1) = 1*0 + 1*0 + 1*1 = 1$

$z2 = 1 / (1 + e^{-n2}) = 1 / (1 + e^{-1}) = 1 / (1 + 0.368) = 0.731$

$z3 = 0.731$

$n1 = w_{12}\, z_2 + w_{13}\, z_3 + w_{14}\,(+1) = 1*.731 + 1*.731 + 1*1 = 2.462$

$z1 = 1 / (1 + e^{-n1}) = 1 / (1 + e^{-2.462}) = 0.921$

Pasos hacia atrás:
Como la neurona 1 es de salida, entonces

$$\delta_1 = \mathbf{Z}_1(1-\mathbf{Z}_1)(\mathbf{T}_1 - \mathbf{Z}_1)$$

$$\delta_1 = 0.921\ (1 - 0.921)\ (1 - 0.921) = 0.00575$$

$$\Delta_{\mathbf{p}}\mathbf{W}_{\mathbf{ji}} = \eta\delta_{\mathbf{pj}}\mathbf{Z}_{\mathbf{pi}}$$

$$\Delta \mathbf{W}_{12} = \eta \delta_1 \mathbf{Z}_2 = 1*0.00575*0.731 = 0.0042$$

$$\Delta \mathbf{W}_{13} = \eta \delta_1 \mathbf{Z}_3 = 1*0.0575*0.731 = 0.0042$$

$$\Delta \mathbf{W}_{14} = \eta \delta_1 \mathbf{Z}_4 = 1*0.00575*1 = 0.00575$$

Las neuronas 2 y 3 son unidades ocultas, entonces $\delta_{\mathbf{pj}} = \mathbf{Z_{pj}}(1 - \mathbf{Z_{pj}})\left(\sum_{\mathbf{k}} \delta_{\mathbf{pk}} \mathbf{w_{kj}} \right)$, donde el rango de *k* es 1 por existir solamente una neurona de salida. Luego,

$\delta_2 = \mathbf{Z}_2(1 - \mathbf{Z}_2)\left(\delta_1 \mathbf{w}_{12} \right)$ = 0.731*(1-0.731)*0.00575*1= 0.00113

$\delta_3 = \mathbf{Z}_3(1 - \mathbf{Z}_3)\left(\delta_1 \mathbf{w}_{13} \right)$ = 0.731*(1-0.731)*0.00575*1= 0.00113

$$\Delta_{\mathbf{p}} \mathbf{w}_{\mathbf{ji}} = \eta \delta_{\mathbf{pj}} \mathbf{Z}_{\mathbf{pi}}$$

$$\Delta \mathbf{w}_{25} = \eta \delta_2 \mathbf{Z}_5 = 1*0.00113*0 = 0$$

$$\Delta \mathbf{w}_{35} = \eta \delta_3 \mathbf{Z}_5 = 1*0.00113*0 = 0$$

$$\Delta \mathbf{w}_{26} = \eta \delta_2 \mathbf{Z}_6 = 1*0.00113*0 = 0$$

$$\Delta \mathbf{w}_{36} = \eta \delta_3 \mathbf{Z}_6 = 1*0.00113*0 = 0$$

$$\Delta \mathbf{w}_{27} = \eta \delta_2 \mathbf{Z}_7 = 1*0.00113*1 = 0.00113$$

$$\Delta \mathbf{w}_{37} = \eta \delta_3 \mathbf{Z}_7 = 1*0.00113*1 = 0.00113$$

De aquí los nuevos pesos después de la primer par de entrenamiento son:

$W_{12} = 1.0042$, $W_{13} = 1.0042$, $W_{14} = 1.00575$

$W_{25} = 1$, $W_{26} = 1$, $W_{27} = 1.00113$

$W_{35} = 1$, $W_{36} = 1$, $W_{37} = 1.00113$

Cuáles serían para el segundo conjunto de entrenamiento?

Desarróllelo……

UN EJEMPLO USANDO BACKPROPAGATION Y MATLAB

1. Neural0.m

```
%
%  BACKPROPAGATION.
%
% 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.
%
%

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.
%
%

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. (neural1)
%


%
% CREA LOS ARCHIVOS EN FORMATO ESTANDAR
%



%======================
 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

% ============================================================
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. (neural1a)
%


%
% CREA LOS ARCHIVOS DE ENTRENAMIENTO
%



%======================
 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


% ========================================================
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)
%

%
% 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)



% PREPARACION DE LA MATRIZ 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

% GUARDANDO LOS PATRONES DE NTRADA Y SALIDA
% ========================================================
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

```
%
%
%  ESCALANDO LOS EJEMPLOS DE ENTRADA Y SALIDA

% DEFINIENDO LOS PATRONES DE ENTRADA DEFINITIVOS
% ==============================================
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. (neural2)
%
%  ESCALAMIENTO DE LOS EJEMPLOS
% ================================================
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. (neural2)
%
%  ESCALAMIENTO DE LOS EJEMPLOS
% ===============================================
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.
%
% 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';

% PATRONES DE ENTRADA
% ====================================

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


% -------------------------------------------------------

number_inputs=M;
number_patterns=N;


%
% INICIALIZA LA ARQUITECTURA DE LA RED
% ==================================
% 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: NUMERO DE NODOS OCULTOS (PUDE SER ACTUALIZADO)

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 GENERADOR DE NOS. ALEATORIOS.
```

```
%  W1=rands(S1,R);
%  B1=rands(S1,1);
%  W2=rands(S2,S1);
%  B2=rands(S2,1);
% NWLOG GENERADOR ALEATORIOS Nguyen-Widrow PARA NEURONAS LOGSIG
  [W1,B1]=nwlog(S1,R);
  [W2,B2]=nwlog(S2,S1);
else
  load whgts.mat
% REEMPLAZA LOS PESOS PORLOS PESOS VIEJOS (PREVIAMENTE GRABADOS)
  W1=NW1; B1=NB1;W2=NW2;B2=NB2;
end
% ---------------------------------------------------------

% ENTRENA LA RED
% ================

% PARAMETROS DE ENTRENAMIENTO

% PARAMETROS
% ERROR DESEADO: error_goal
% MAXIMO NUMERO DE CICLOS: max_epochs
% TASA DE APRENDIZAJE: lr
%
% ESTOS PARAMETROS PUEDEN SER CAMBIADOS

  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';

% MUESTRA LOS PARAMETROS DE DEFINICIÓN

  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

% ENTRADA DE LOS NUEVOS PARAMETROS


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


% FASE DE ENTRENAMIENTO

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);


% GRAFICA DE LA CURVA DEL ERROR
%==============================
figure;
ploterr(error);
pause


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

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

Neural4.m

```
%
%  BACKPROPAGATION. (neural4)
%
%  USA ouput.mat Y input.mat COMO MATRICES DE PRUEBA

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

load TestOut;
load TestInp;



% =======================================

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



% -------------------------------------------------------

number_inputs=M;
number_patterns=N;



% =======================================

P=Inp';
T=Out';

clear Inp; clear Out;

% COMPARACAION DE LAS SALIDAS DESEADAS Y CALCULADAS
% -------------------------------------------------------

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


%
% ==============================
%


load whgts.mat

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

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

```
% PRUEBA LA RED
% =============


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. (neural5)
%
% GUARDA LOS EJEMPLOS DE ENTRADA Y SALIDA COMO EJEMPLOS DE ENTRENAMIENTO
% Y DE PRUEBA

%

% ================================================

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;
```

LA VARIACION MOMENTO EN BACKPROPAGATION

Mejora los tiempos de entrenamiento a backpropagation.

En lugar de calcular los pesos para un patrón de entrenamiento mediante $\Delta_{\mathbf{p}}\mathbf{w}_{\mathbf{ji}} = \eta\delta_{\mathbf{pj}}\mathbf{Z}_{\mathbf{pi}}$, se le añade un término adicional dado por una contribución proporcional del término momento $\alpha\cdot$y el cambio de los pesos anteriores. Es decir,

$$\Delta_{\mathbf{p}}\mathbf{w}_{\mathbf{ji}} = \eta\delta_{\mathbf{pj}}\mathbf{Z}_{\mathbf{pi}} + \alpha\cdot\Delta_{\mathbf{p}}\mathbf{w}_{\mathbf{ji}}(anterior),$$

el valor de $\alpha\cdot$debe estar entre 0 y 1.

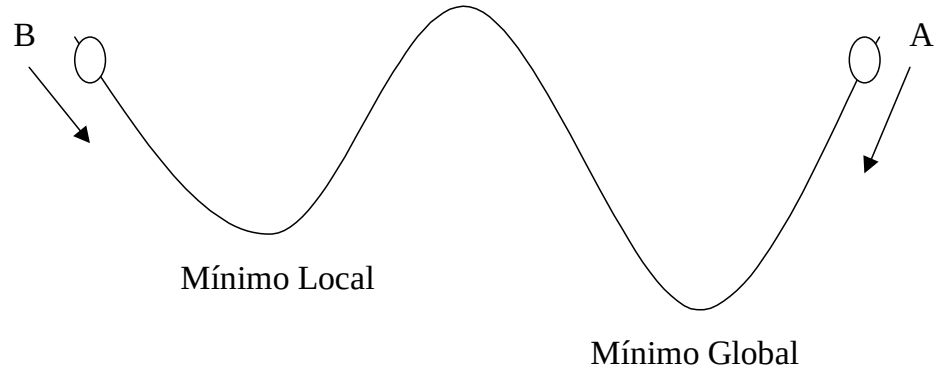
Con respecto a esta variación, se ha encontrado que en algunos casos puede producir buenos resultados cuando alcanza un área plana o una profundidad estrecha y bastante grande. En estos casos valores cercano a uno (por ej.: 0.9) podrían ayudar a mejorar la solución.

En otros casos, por ejemplo áreas ondulantes amplias y pocas profundas, no ayudarían en mayor grado.

De ahí que la solución pueda estar afectada por varios mínimos locales, haciendo inalcanzable un mínimo global.

ALGUNOS COMENTARIOS SOBRE BACKPROPAGATION

La Regla Delta (generalizada) empleada por el algoritmo backpropagation puede decrecer el error cuadrático. Sin embargo puede conducir a un mínimo local más que a un mínimo global.



Está establecido que el mínimo local existe. Sin embargo, no se sabe si ellos están cerca de los mínimos globales.

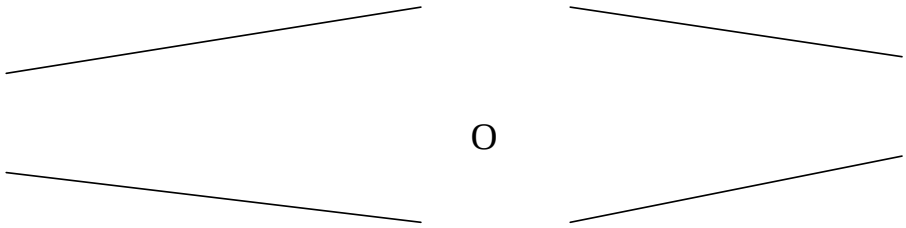
La superficie del error puede presentar muchas áreas planas donde los cambios de los pesos son mínimos y no ayudan a conseguir los mínimos globales. De ahí, los mecanismos de ayuda al algoritmo original.

Actualmente se dispone de variantes numéricas que contribuyen a mejorar la eficiencia del algoritmo original. Tal es el caso de los algoritmos del gradiente conjugado, Levenberg-Marquardt, Quasi-Newton, entre otros, ya disponibles en software especializado. (Matlab, Statistica Neural Networks, etc)

No existe una teoría escrita para guiarnos en como seleccionar el número de capas, número de nodos o neuronas por capas. Sin embargo, en la actualidad existen métodos alternativos para mejorar

la eficiencia del algoritmo, tales como el uso de los algoritmos genéticos para establecer a priori un diseño de la red, análisis de regresión como recurso de preprocesamiento para conocer los pesos iniciales, transformación de variables, etc.

> La regla del dedo que determina que el promedio de la suma del número de nodos de entrada y de salida indicaría un valor inicial del número de nodos ocultos. Algo como un diseño de trompeta en ambos sentidos.



> Teóricamente una capa oculta es suficiente para capturar la no linealidad del problema. Simon Haykin recomienda dos capas: la primera para capturar las características locales y la segunda para las características globales.

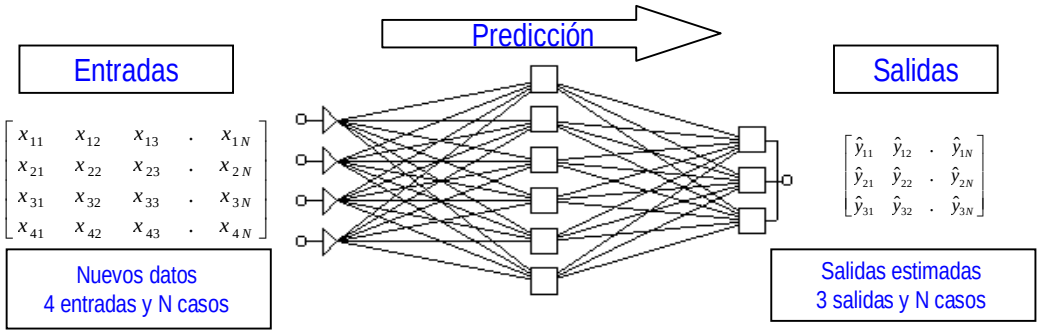
Una de las principales dificultades en este tipo de redes es el tiempo de entrenamiento. En ocasiones, son demasiados extensos como producto de la complejidad del diseño de la red neuronal.

La selección del valor de la constante de aprendizaje a veces resulta un arte.

La especificidad aplicativa de un modelo de redes neuronales justifica más aún el cuidado que se debe tener en las fases de preparación y diseño del modelo. Tal es el caso del pre-procesamiento de los datos y la capacidad de generalización alcanzada por el modelo.

ALGUNOS COMENTARIOS SOBRE GENERALIZACION EN BACKPROPAGATION

En una RN entrenada, si las salidas calculadas por la red con nuevos ejemplos están próximas a los valores deseados, hay generalización (Haykin, 1995).



De este modo, entendamos por generalización, la habilidad de una red neuronal de almacenar en sus pesos sinápticos características que le son comunes a todos los patrones de entrenamiento que fueron usados durante la fase de entrenamiento.

Cuando una nueva entidad es almacenada, las fortalezas de las conexiones existentes representadas por los pesos se modifican suavemente. Si estas modificaciones refuerzan los pesos previamente almacenados entonces ocurre generalización.

Las entradas a la RN durante el entrenamiento deben contener suficiente información perteneciente al dominio. Una buena generalización ocurre cuando los ejemplos de entrenamiento son los suficientemente representativos del dominio del cual forman parte.

El número de casos utilizados para el entrenamiento deben ser suficientes y además representativos del conjunto de casos que se desea generalizar

Si el problema es de clasificación, los ejemplos que formen parte del entrenamiento deben representar todas las categorías. La red puede aprender mejor con una mayor cantidad de ejemplos de todas las categorías.

Si es de aproximación (vale para predicción), se deben incluir ejemplos de valores continuos a través de todo el rango. Es decir, de la nube de ejemplos de entrenamiento, la muestra debe contener ejemplos de los bordes y del centro para un mejor entrenamiento.

Estas dos características principales vigiladas durante el entrenamiento resultan vitales para lograr una buena generalización.

Los patrones generalizados son la suma de todas las modificaciones separadas que han ocurrido. Es decir, en la medida que un patrón de entrenamiento es presentado a la red, ellos pueden generalizar debido al cambio ocurrido en sus pesos.

RECOMENDACIONES PARA EL ENTRENAMIENTO Y GENERALIZACION DE UNA RED EN BACKPROPAGATION

Existen cuatro fases fundamentales en la construcción de modelo de red neuronal. A continuación se mencionarán algunas sugerencias que podrían ser tomadas en cuenta cuando se intente el desarrollo completo de un modelo. Estas sugerencias son válidas desde cualquier aplicación que se emplee como herramienta de desarrollo. Sin embargo, alguna de estas sugerencias son resueltas ya desde algunas aplicaciones. Por ejemplo desde Matlab, Predict pueden hacerse ensayos previos de la arquitectura de la red.

Arquitectura:

Como ya se mencionó, un comienzo de construcción puede ser el de asumir que el número de neuronas en la capa oculta sean menos de la mitad de la suma del número de entradas más el de las salidas.

Si no es suficiente porque el entrenamiento es pobre, incrementar con pocas unidades y proceder a entrenar nuevamente.

Este proceso se puede continuar hasta tanto no se observe mejoras en las pruebas del entrenamiento.

Sin embargo, hay que mantener clara la siguiente premisa: “La mejor red neuronal entrenada no es aquella cuyos error continuamente está bajando”. Esto puede conducir a memorización.
Existe una relación donde se puede determinar la cantidad de ejemplos de entrenamiento necesarios para un error deseado y un tamaño de red establecido. Esto es, $N > W/E$, donde N es la cantidad de ejemplos de entrenamiento, W la cantidad de nodos ocultos y E el error deseado.

Entrenamiento:

Después de haber definido una buena arquitectura para la red, en esta fase, se deben atender algunos pasos que son importantes para un buen desempeño del entrenamiento.

El primero de ello es la inicialización de la red. El inicio del entrenamiento requiere de que los pesos tengan un valor inicial. Varios mecanismos existen para la determinación de estos pesos. De manera aleatoria, por aproximación mediante regresión múltiple son algunos de los métodos empleados. Sin embargo, si el proceso de entrenamiento fue iniciado y deba ser suspendido, no se requiere que

la red sea entrenada desde sus comienzos. Los últimos pesos serían los pesos iniciales cuando se desee continuar el entrenamiento con nuevos grupos de datos de entrenamiento.
La constante de aprendizaje juega un importante papel durante la definición de los parámetros para el entrenamiento. Por ejemplo, en Matlab este valor puede flotar en el sentido que en la medida que requiera cambios, el algoritmo automáticamente lo altera a los fines de darle una mayor eficiencia al entrenamiento medido a través de la minimización del error cuadrático. Sin emabargo, se repite la premisa: “La mejor red neuronal entrenada no es aquella cuyos error continuamente está bajando”.

Validación:

Es conveniente añadir ruido a los pesos a través de ejemplos que permitan continuar el proceso cuando se conoce que el entrenamiento ha caído en un mínimo local.

Se debe repetir este proceso hasta conseguir que los parámetros sean satisfechos:
error deseado siguiendo un patrón de caída aceptable,
tiempos de entrenamiento no muy extensos. El tiempo crece exponencialmente de acuerdo a número de entradas.

Guardar al menos el 10% de los datos (entradas y salidas) para hacer pruebas de la eficiencia de la red.

Generalización del modelo:

El mejor desempeño que pueda tener el modelo recién entrenado es que cuando sean presentados ejemplos nunca vistos por la red, pueda clasificarlos o aproximarlos con el mínimo error ajuste.

Un mal ajuste de estos nuevos ejemplos y un entrenamiento supuestamente bueno, implica una memorización de los patrones de entrenamiento.

Por tanto, se debe estar seguro de que no existan contradicciones en los datos y para ello se debe vigilar la preparación de los datos como parte de los recursos para un buen desempeño de una red.