

```
1 % Proyecto de Grado
2 % Red Neuronal Evolutiva
3 % Juan Sebastian Henao Parra
4 % Version 4.2
5
6 %% PREPARA EL ESPACIO DE TRABAJO PARA CALCULOS
7
8 %Carpeta
9 %cd('C:\Users\Sebastian\SkyDrive\Proyecto de Grado\source\source 4.0 (13-03-24)')
10
11 %Inicializacion
12 clear all; close all; clc
13
14 %% CREA PARAMETROS QUE SE UTILIZARAN
15
16 global datos columnaSerie columnaDesempeno tamanoHorizontes numeroHorizontes
17
18 %Definicion de tamaño de los horizontes y el numero horizontes
19 tamanoHorizontes=100;
20 numeroHorizontes=10;
21
22 %Definicion del periodo a entrenar
23 periodoEntrenamiento=0.50;
24
25 %Definicion de la columna cargada que se usara y el numero de rezagos
26 columnaSerie=3;
27 rezagosParteLineal=[1 4 12 24 29];
28 rezagosParteNoLineal=[1 4 12 24 29];
29
30 %Definicion de la topologia
31 numeroCapasOcultas=1;
32 numeroNeuronas=4;
33
34 %Definicion numero de busquedas de la red neuronal artificial
35 numeroBusquedas=30;
36
37 %Definicion numero de neuroevolucion
38 numeroNeuroEvolucion=1;
39
40 %Tipo de funcion de costo
41 % 1 Para Cuadratica
42 % 2 Para Absoluta
43 % 3 Para LINEX
44 % 4 Para Absoluto Asimetrico
45 tipoFuncion=1;
46
47 %Tipo de normalizacion
48 % 1 Para MIN y MAX
49 % 2 Para Media y Desviacion Estandar
50 tipoNormalizacion=1;
51
52 %Algoritmo de optimizacion
53 % 1 Para SIMPLEX
54 % 2 Para BFGS
55 % 3 Para Stepest Decent
```

```

56 algoOptim=2;
57
58 %Definicion de archivos para los resultados
59 seccionANN=0:numeroBusquedas:numeroBusquedas*(numeroNeuronas-1);
60 seccionRezagosANN=0:numeroBusquedas*numeroNeuronas:numeroBusquedas*numeroNeuronas*↖
(numeroRezagosParteNoLineal,2)-1);
61 numeroANNS=numeroBusquedas*numeroNeuronas*size(rezagosParteNoLineal,2);
62 seccionENN=0:numeroNeuroEvoluciones:numeroNeuroEvoluciones*(numeroNeuronas-1);
63 seccionRezagosENN=0:numeroNeuroEvoluciones*numeroNeuronas:↖
numeroNeuroEvoluciones*numeroNeuronas*(size(rezagosParteNoLineal,2)-1);
64 numeroENNS=numeroNeuroEvoluciones*numeroNeuronas*size(rezagosParteNoLineal,2);
65
66 %% CREA LOS MONITORES
67
68 multiWaitbar( 'Calculando Horizontes ANN', 1/4, 'CancelFcn', @(a,b) disp( ['Cancel',a] ) );
69 multiWaitbar( 'Calculando ANN', 2/4, 'CancelFcn', @(a,b) disp( ['Cancel ',a] ) );
70 multiWaitbar( 'Calculando Horizontes ENN', 3/4, 'CancelFcn', @(a,b) disp( ['Cancel',a] ) );
71 multiWaitbar( 'Calculando ENN', 4/4, 'CancelFcn', @(a,b) disp( ['Cancel ',a] ) );
72 multiWaitbar( 'Calculando Horizontes ANN', (0/(numeroHorizontes)));
73 multiWaitbar( 'Calculando ANN', (0/(numeroANNS)));
74 multiWaitbar( 'Calculando Horizontes ENN', (0/(numeroHorizontes)));
75 multiWaitbar( 'Calculando ENN', (0/(numeroENNS)));
76 h(1) = uicontrol↖
('String','Pause','Callback','uiwait','units','Normalized','Position',[0,0, 0.5, 1]);
77 h(2) = uicontrol↖
('String','Resume','Callback','uiresume','units','Normalized','Position',[0.5,0, 0.5, ↆ
1]);
78
79 %% CARGA ARCHIVOS CON DATOS
80 datos=xlsread('Datos','COL20');% Carga el archivo y la hoja mencionada
81
82 %% RED NEURONAL ARTIFICIAL
83
84 for numHorizonte=1:numeroHorizontes
85
86     for numRezagosParteNoLineal=1:size(rezagosParteNoLineal,2)
87         rezagosParteNoLinealTemp=rezagosParteNoLineal(:,1:numRezagosParteNoLineal);
88
89         for numNeurona=1:numeroNeuronas
90
91             for numBusqueda=1:numeroBusquedas
92
93                 % CREA TEMPORALMENTE LAS DIFERENTES MATRICES
94                 [dentroMuestraNoLineal, fueraMuestraNoLineal,...]
95                     dentroMuestraLineal, fueraMuestraLineal]=generarInput...
96                     (rezagosParteNoLinealTemp,rezagosParteLineal,numHorizonte);
97                     [dentroMuestraNoLinealN, fueraMuestraNoLinealN,...]
98                     dentroMuestraLinealN, fueraMuestraLinealN]↖
=generarInputNormalizado...
99                     (rezagosParteNoLinealTemp,rezagosParteLineal,numHorizonte,↖
tipoNormalizacion);
100
101             % GENERA PESOS INICIALES, PHI, BETHA y ALPHA O TRAE LOS DE

```

```

102      % HORIZONTES ANTERIORES
103      if numHorizonte==1
104          [parametros]=generarPesosIniciales(numeroCapasOcultas, ...
105              numNeurona, rezagosParteNoLinealTemp, rezagosParteLineal, ...
106              dentroMuestraNoLinealN,dentroMuestraLinealN);
107      else
108          cuboDatosTemp=cubosDatos (:,:,numRezagosParteNoLineal,↖
numHorizonte-1);
109          [parametros,topologia]=generarPesosRolling...
110              (cuboDatosTemp,numBusqueda+seccionANN(1,numNeurona));
111      end
112
113          fullConected=ones(size(rezagosParteNoLinealTemp,2)↖
+numeroCapasOcultas+1,numNeurona);
114          llave=generarLlave(rezagosParteLineal,rezagosParteNoLinealTemp,↖
numNeurona);
115
116          % ENCUENTRA LOS PARAMETROS QUE MINIMIZAN LA FUNCION DE ERROR
117          %options = optimset('Display','iter','TolFun',1e-8,'PlotFcns',↖
@optimplotfval,'HessUpdate','bfgs','LargeScale','off');
118          options=optimset↖
('HessUpdate','bfgs','LargeScale','off','MaxFunEvals',1000000,'MaxIter',10000);
119          [x,fval]=fminunc(@(parametros) funcionCosto(parametros, ...
120              dentroMuestraNoLinealN,dentroMuestraLinealN,tipofuncion, ...
121              llave,numNeurona),parametros,options);
122
123          % GUARDA LA INFORMACION EN UN CUBO
124          [phi,betha,alpha]=separarMatrices(x,llave,numNeurona);
125          [rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM] = ...
126              obtenerDesempeno(x,dentroMuestraLinealN,↖
dentroMuestraNoLinealN, ...
127                  fueraMuestraLinealN,fueraMuestraNoLinealN,fullConected, ...
128                  llave,numNeurona);
129          cubosDatos(numBusqueda+seccionANN(1,numNeurona),:,:,↖
numRezagosParteNoLineal, ...
130              numHorizonte)=[numBusqueda+seccionANN(1,numNeurona),↖
numBusqueda,numNeurona, ...
131              numRezagosParteNoLineal,numHorizonte,x,{fval},{phi},{betha},↖
{alpha},{fullConected}, ...
132                  {rezagosParteLineal},{rezagosParteNoLinealTemp},rmseDM, ...
133                  rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM];
134
135          % GENERA UNA VENTANA PARA MONITOREAR EL AVANCE DEL PROGRAMA
136          abort = multiWaitbar( 'Calculando ANN', (numBusqueda+seccionANN(1, ...
137              numNeurona)+seccionRezagosANN(1,numRezagosParteNoLineal))/(numeroANNS));
138          if abort
139              break
140          else
141              pause( 1 )
142          end
143      end
144  end
145
146 end

```

```

147
148     abort = multiWaitbar( 'Calculando Horizontes ANN', (numHorizonte/↙
149 (numeroHorizontes)));
150     if abort
151         break
152     else
153         pause( 1 )
154     end
155 end
156
157 %% RED NEURONAL EVOLUTIVA
158
159 for numHorizonte=1:numeroHorizontes
160
161     for numRezagosParteNoLineal=1:size(rezagosParteNoLineal,2)
162         rezagosParteNoLinealTemp=rezagosParteNoLineal(:,1:numRezagosParteNoLineal);
163
164         for numNeurona=1:numeroNeuronas
165
166             for numNeuroEvolucion=1:numeroNeuroEvoluciones
167
168                 % CREA TEMPORALMENTE LAS DIFERENTES MATRICES
169                 [dentroMuestraNoLineal, fueraMuestraNoLineal, ↵
dentroMuestraLineal,...]
170                         fueraMuestraLineal]=generarInput(rezagosParteNoLinealTemp,...,
171                                         rezagosParteLineal,numHorizonte);
172                         [dentroMuestraNoLinealN, fueraMuestraNoLinealN, ↵
dentroMuestraLinealN,...]
173                         fueraMuestraLinealN]=generarInputNormalizado↖
(rezagosParteNoLinealTemp,...,
174                                         rezagosParteLineal,numHorizonte, tipoNormalizacion);
175
176             % GENERA PESOS INICIALES, PHI, BETHA Y ALPHA O TRAE LOS DE
177             % HORIZONTES ANTERIORES
178             if numHorizonte==1
179                 [parametrosIniciales]=generarPesosIniciales↖
(numeroCapasOcultas,...)
180                     numNeurona,rezagosParteNoLinealTemp,rezagosParteLineal,...,
181                     dentroMuestraNoLinealN,dentroMuestraLinealN);
182                     [topologiaInicial]=unirTopologia(ones(size↖
(rezagosParteNoLinealTemp,2)+...
183                                         numeroCapasOcultas+1,numNeurona));
184             else
185                 cuboDatosTemp=cubosDatosNeuroEvolucion(:, :, ↵
numRezagosParteNoLineal,numHorizonte-1);
186                 [parametrosIniciales,topologiaInicial]=generarPesosRolling...
187                               (cuboDatosTemp,numNeuroEvolucion+seccionENN(1,numNeurona))
188             end
189
190             fullConected=ones(size(rezagosParteNoLinealTemp,2)+...
191                             numeroCapasOcultas+1,numNeurona);
192             llave=generarLlave(rezagosParteLineal,rezagosParteNoLinealTemp,...,
193                             numNeurona);
194

```

```

195          % NEUROEVOLUCION
196          [parametrosMejorTopologia,mejorTopologia,costoMejorTopologia,...  

197              ultimaGeneracion,cuboDesempeno]=algoritmoEvolutivo ↵  

198 (parametrosIniciales,...  

199             topologiaInicial,fullConected,numeroCapasOcultas, ↵  

200 rezagosParteNoLinealTemp,...  

201             rezagosParteLineal,dentroMuestraNoLinealN, ↵  

202 dentroMuestraLinealN,...  

203             fueraMuestraNoLinealN,fueraMuestraLinealN, ↵  

204 dentroMuestraNoLineal,...  

205             dentroMuestraLineal,fueraMuestraNoLineal,fueraMuestraLineal, ↵  

206 tipoFuncion,numNeurona);  

207  

208          %GUARDAR LA INFORMACION EN UN CUBO  

209          [phi,betha,alpha]=separarMatrices (parametrosMejorTopologia,...  

210              llave,numNeurona);  

211          [rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM] =...  

212              obtenerDesempeno (parametrosMejorTopologia, ↵  

213 dentroMuestraLinealN,...  

214             dentroMuestraNoLinealN,fueraMuestraLinealN, ↵  

215 fueraMuestraNoLinealN,...  

216             mejorTopologia,llave,numNeurona);  

217          cubosDatosNeuroEvolucion (numNeuroEvolucion+seccionENN(1, ↵  

218 numNeurona),:,numRezagosParteNoLineal,...  

219             numHorizonte)=[numNeuroEvolucion+seccionENN(1,numNeurona), ↵  

220 numNeuroEvolucion,numNeurona,...  

221             numRezagosParteNoLineal,numHorizonte, ↵  

222 parametrosMejorTopologia,...  

223             {costoMejorTopologia},{phi},{betha},{alpha}, ↵  

224 {mejorTopologia},...  

225             {rezagosParteLineal},{rezagosParteNoLinealTemp},...  

226             rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM];  

227  

228          % GENERA UNA VENTANA PARA MONITOREAR EL AVANCE DEL PROGRAMA  

229          abort = multiWaitbar( 'Calculando ENN', ↵  

230 (numNeuroEvolucion+seccionENN(1,numNeurona)+seccionRezagosENN(1, ↵  

231 numRezagosParteNoLineal))/(numeroENNS));  

232          if abort  

233              % Here we would normally ask the user if they're sure  

234              break  

235          else  

236              pause( 1 )  

237          end  

238  

239      end  

240  

241      abort = multiWaitbar( 'Calculando Horizontes ENN', (numHorizonte/ ↵  

242 (numeroHorizontes)));  

243      if abort  

244          break  

245      else

```

```
236         pause( 1 )
237     end
238
239 end
240
241 multiWaitbar( 'CloseAll' );
242
243 %% OBTIENE LAS MEJORES REDES Y LA MEJOR RED
244 %Las columnas de desempeño pueden ser 18-19-20-21
245 numHorizonteEval=10;
246 columnaDesempeno=20;
247
248 for numHorizonteEval=1:10
249 [mejorANN,matrizDatos] = obtenerMejorRed(size(rezagosParteNoLineal,2),cubosDatos
(:, :, :, numHorizonteEval));
250 mejorANNHorizonte(numHorizonteEval,:) = mejorANN
251 [mejorENN,matrizDatosNeuroEvolucion] = obtenerMejorRed(size(rezagosParteNoLineal,
2),cubosDatosNeuroEvolucion(:, :, :, numHorizonteEval));
252 mejorENNHorizonte(numHorizonteEval,:) = mejorENN
253 end
254
255 %[mejorANN,matrizDatos] = obtenerMejorRed(size(rezagosParteNoLineal,2),cubosDatos
(:, :, :, numHorizonteEval));
256 %[mejorENN,matrizDatosNeuroEvolucion] = obtenerMejorRed(size(rezagosParteNoLineal,
2),cubosDatosNeuroEvolucion(:, :, :, numHorizonteEval));
257
258 %% GUARDA LOS DATOS EN UN ARCHIVO
259 formatOut = 'mm-dd-yy';
260 datestr(now,formatOut);
261 filenameNN = sprintf('Datos guardados con %d neuronas el', numeroNeuronas);
262 filenameDD = sprintf(datestr(now,formatOut));
263 filename = strcat(filenameNN, '_', filenameDD);
264 save(filename);
265
266 %% GENERA GRAFICAS
267 generarGraficas(mejorANN,mejorENN,rezagosParteNoLineal,rezagosParteLineal,
numHorizonte)
268
```

```
1 function [topologia] = unirTopologia(topologia)
2 %OBJETIVO: Unir la matriz de la topología recomponiéndola.
3 %COMPORTAMIENTO: El comando reshape la une en una sola fila.
4 %RETORNA: Una fila con la topología inicial.
5
6 topologiaTemp=topologia';
7
8 topologia=[reshape(topologiaTemp,1,[])];
9
10 end
11
12
```

```
1 function [parametros] = unirMatrices(phi,betha,alpha)
2 %OBJETIVO: Unir las matrices phi, alpha y betha para ser introducidas
3 %en la funcion de costo.
4 %COMPORTAMIENTO: El comando reshape las une en una sola fila.
5 %RETORNA: Una fila con los parametros.
6
7 alpha=alpha';
8
9 parametros=[reshape(phi,1,[]) reshape(alpha,1,[]) reshape(betha,1,[]) ];
10
11 end
12
13
```

```
1 function g = sigmoid(z)
2 %OBJETIVO: Funcion Logistica.
3 %COMPORTAMIENTO: Evalua.
4 %RETORNA: Valor de z evaluado en la funcion logistica.
5
6 g = 1.0 ./ (1.0 + exp(-z));
7
8 end
9
```

```
1 function [topologia] = separarTopologia(poblacion,llave,numeroNeuronas)
2 %OBJETIVO: Separar la topologia requerida retornando una matriz.
3 %COMPORTAMIENTO: Convierte la fila poblacion en una matriz.
4 %RETORNA: La matriz con la topologia.
5
6 numeroEntradas=llave(1,3)+1;
7
8 poblacion=poblacion';
9 b=reshape(poblacion,numeroNeuronas,numeroEntradas);
10 topologia=b';
11
12 end
13
14
```

```
1 function [phi,betha,alpha] = separarMatrices(parametros,llave,numeroNeuronas)
2 %OBJETIVO: Separar las matrices phi, alpha y betha las cuales representan
3 %los pesos de cada una de las conecciones.
4 %COMPORTAMIENTO: Convierte la fila parametros en 3 matrices.
5 %RETORNA: La matriz phi parte lineal, la matriz alpha pesos de las neuronas
6 % y betha que son los pesos sinapticos.
7
8 numeroEntradas=llave(1,3);
9 numeroNeuronas;
10 phi=parametros(:,1:llave(1,1));
11
12 matrizTemp=parametros(:,llave(1,1)+1:end);
13
14 matrizTemp=matrizTemp';
15 b=reshape(matrizTemp,numeroNeuronas,numeroEntradas+1);
16 matrizTemp=b';
17
18 alpha=matrizTemp(1:end-1,:);
19 betha=matrizTemp(end:end,:);
20
21 end
22
23
```

```
1 function [mejorRed,matrizDatos] = obtenerMejorRed(numeroRezagos,cuboDatos)
2 %OBJETIVO: Obtener la mejor por cada rezago incluido y guardarla en una matriz para ↵
luego
3 %obtener la mejor, esto se asemeja a una reduccion de dimensiones.
4 %COMPORTAMIENTO: Las mejores por cada pagina son capturadas y puestas en
5 %matrizDatos donde nuevamente se organizara y se obtendra la mejor.
6 %RETORNA: Una matriz con la mejor red por cada rezago adicional devuelve
7 %la mejor red.
8
9 global datos columnaSerie columnaDesempeno tamanoHorizontes numeroHorizontes
10
11 for numRezago=1:numeroRezagos
12     matrizTemp=cuboDatos(:,:,numRezago);
13     desempenoRezago=sortrows(matrizTemp,columnaDesempeno);
14     matrizDatos(numRezago,:)=desempenoRezago(1,:);
15 end
16
17 matrizDatos = sortrows(matrizDatos,columnaDesempeno);
18 mejorRed=matrizDatos(1,:);
19
20 end
21
22
```

```
1 function [parametros] = obtenerMejoresParametros(cuboDatos,numeroHorizonte,  
numeroRezagos,numeroNeuronas)  
2 %OBJETIVO: Obtener y unir los parametros de la mejor red de dicho horizonte.  
3 %COMPORTAMIENTO: Del cubo de datos se extraen .  
4 %RETORNA: Una matriz con la mejor red por cada rezago adicional devuelve  
5 %la mejor red.  
6  
7 global columnaDesempeno  
8  
9 cuboFiltrado=cuboDatos(:,:,:numeroRezagos,numeroHorizonte);  
10 cuboFiltrado = cuboFiltrado(cell2mat(cuboFiltrado(:, 3)) == numeroNeuronas,:);  
11 mejorRed=sortrows(cuboFiltrado,columnaDesempeno);  
12 mejorRed=mejorRed(1,:);  
13 phi=cell2mat(mejorRed(1,6));  
14 betha=cell2mat(mejorRed(1,7));  
15 alpha=cell2mat(mejorRed(1,8));  
16  
17 [parametros] = unirMatrices(phi,betha,alpha);  
18  
19 end  
20  
21
```

```

1 function [rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM] =↵
obtenerDesempeno(parametros,dentroMuestraLinealN,dentroMuestraNoLinealN,↵
fueraMuestraLinealN,fueraMuestraNoLinealN,topologia,llave,numeroNeuronas)
2 %OBJETIVO: Obtener el desempeño dentro y fuera de muestra de unos parametros y una
3 % topología dada.
4 %COMPORTAMIENTO: Son capturadas las matrices que contienen los datos, los parametros ↵
y
5 %las neuronas.
6 %RETORNA: Los valores rmseDM, rmspeDM, maeDM, mapeDM, rmseFM, rmspeFM, maeFM, mapeFM.
7
8 %Obtiene el desempeño calculando las medidas
9 [phi,betha,alpha]=separarMatrices(parametros,llave,numeroNeuronas);
10
11 if size(topologia,1)==1
12 topologia=separarTopologia(topologia,llave,numeroNeuronas);
13 end
14
15 %Ajuste
16 A_1_AD=dentroMuestraNoLinealN(:,2:end);
17 Z_2_AD=A_1_AD*(topologia(1:end-1,:).*alpha);
18 A_2_AD=sigmoid(Z_2_AD);
19 A_3_AD=(dentroMuestraLinealN(:,2:end)*phi')+(A_2_AD*((topologia(end:end,:))'.*↵
*betha')));
20
21 %Pronostico
22 A_1_FO=fueraMuestraNoLinealN(:,2:end);
23 Z_2_FO=A_1_FO*(topologia(1:end-1,:).*alpha);
24 A_2_FO=sigmoid(Z_2_FO);
25 A_3_FO=(fueraMuestraLinealN(:,2:end)*phi')+(A_2_FO*((topologia(end:end,:))'.*↵
*betha')));
26
27 %%%%%%%%%%%%%% ESPACIO PARA DESEMPEÑO %%%%%%%%%%%%%%
28
29 nDM=size(dentroMuestraLinealN,1);
30 nFM=size(fueraMuestraLinealN,1);
31
32 rmseDM=sqrt((1/nDM)*sum((dentroMuestraNoLinealN(:,1)-A_3_AD).^2));
33 rmspeDM=sqrt((1/nDM)*sum(((dentroMuestraNoLinealN(:,1)-A_3_AD).↵
/dentroMuestraNoLinealN(:,1)).^2));
34 maeDM=(1/nDM)*sum(abs(dentroMuestraNoLinealN(:,1)-A_3_AD));
35 mapeDM=(1/nDM)*sum(abs((dentroMuestraNoLinealN(:,1)-A_3_AD)./dentroMuestraNoLinealN↵
(:,1)));
36
37 rmseFM=sqrt((1/nFM)*sum((fueraMuestraNoLinealN(:,1)-A_3_FO).^2));
38 rmspeFM=sqrt((1/nFM)*sum(((fueraMuestraNoLinealN(:,1)-A_3_FO)./fueraMuestraNoLinealN↵
(:,1)).^2));
39 maeFM=(1/nFM)*sum(abs(fueraMuestraNoLinealN(:,1)-A_3_FO));
40 mapeFM=(1/nFM)*sum(abs((fueraMuestraNoLinealN(:,1)-A_3_FO)./fueraMuestraNoLinealN(:,1)));
41
42 end
43
44

```

```

1 function [dentroMuestraNoLinealN, fueraMuestraNoLinealN, dentroMuestraLinealN, ↵
fueraMuestraLinealN]=generarInputNormalizado(rezagosParteNoLineal,rezagosParteLineal, ↵
numHorizonte, tipoNormalizacion)
2 %OBJETIVO: Generar las matrices de diseño normalizadas necesarias.
3 %COMPORTAMIENTO: Son capturadas las matrices donde se almacenan los rezagos de la ↵
parte
4 %lineal como de la no lineal, los datos y el numero de la columna que se usara.
5 %RETORNA: Las matrices de diseño necesarias.
6
7 global datos columnaSerie columnaDesempeno tamanoHorizontes numeroHorizontes
8
9 %Obtener los valores que necesita la funcion
10 serie=datos(:,columnaSerie);
11 maximo=max(serie);
12 minimo=min(serie);
13 media=mean(serie);
14 desvest=std(serie);
15
16 %Normalizar todos los datos dependiendo del tipo de normalizacion que escoja
17 if tipoNormalizacion==1
18     for i=1:size(serie,1)
19         serieNormalizada(i,1)=(serie(i,1)-minimo) / (maximo-minimo);
20     end
21 end
22
23 if tipoNormalizacion==2
24     for i=1:size(serie,1)
25         serieNormalizada(i,1)=(serie(i,1)-media) / (desvest);
26     end
27 end
28
29 %Obtener los valores que necesita la funcion
30 serie=datos(:,columnaSerie);
31 numRezagosParteNoLineal=size(rezagosParteNoLineal,2);
32 numRezagosParteLineal=size(rezagosParteLineal,2);
33 numPeriodos=size(datos,1);
34 numDentroMuestra=numPeriodos-tamanoHorizontes-(numeroHorizontes-numHorizonte);
35 numFueraMuestra=numPeriodos-(numeroHorizontes-numHorizonte);
36 nuevosDatosParteNoLineal=zeros(numPeriodos,numRezagosParteNoLineal);
37 nuevosDatosParteLineal=zeros(numPeriodos,numRezagosParteLineal);
38
39 %Remplazar los valores dentro de la matriz vacia de la parte no lineal
40 for i=1:numRezagosParteNoLineal
41     for m=1:numPeriodos-rezagosParteNoLineal(1,i)
42         nuevosDatosParteNoLineal(m+rezagosParteNoLineal(1,i),i)=serieNormalizada(m);
43     end
44 end
45
46 %Remplazar los valores dentro de la matriz vacia de la parte lineal
47 for i=1:numRezagosParteLineal
48     for m=1:numPeriodos-rezagosParteLineal(1,i)
49         nuevosDatosParteLineal(m+rezagosParteLineal(1,i),i)=serieNormalizada(m);
50     end
51 end
52

```

```
53 nuevosDatosParteNoLineal=[serieNormalizada(1:numFueraMuestra,:)
nuevosDatosParteNoLineal(1:numFueraMuestra,:)];
54 nuevosDatosParteLineal=[serieNormalizada(1:numFueraMuestra,:)] nuevosDatosParteLineal
(1:numFueraMuestra,:);
55
56 % Crear 4 matrices de ceros para llenarlas con lo nuevos datos
57 dentroMuestraNoLinealN=nuevosDatosParteNoLineal(1:numDentroMuestra,:);
58 fueraMuestraNoLinealN=nuevosDatosParteNoLineal(numDentroMuestra+1: numFueraMuestra,:);
59 dentroMuestraLinealN=nuevosDatosParteLineal(1:numDentroMuestra,:);
60 fueraMuestraLinealN=nuevosDatosParteLineal(numDentroMuestra+1:numFueraMuestra,:);
61
62 %Organizar las matrices y les agrego el intercepto
63 dentroMuestraNoLinealN= [dentroMuestraNoLinealN(:,1) ones(size
(dentroMuestraNoLinealN,1),1) dentroMuestraNoLinealN(:,2:end)];
64 dentroMuestraLinealN= [dentroMuestraLinealN(:,1) ones(size(dentroMuestraLinealN,1),
1) dentroMuestraLinealN(:,2:end)];
65 fueraMuestraNoLinealN= [fueramuestraNoLinealN(:,1) ones(size(fueramuestraNoLinealN,
1),1) fueraMuestraNoLinealN(:,2:end)];
66 fueraMuestraLinealN= [fueramuestraLinealN(:,1) ones(size(fueramuestraLinealN,1),1)
fueramuestraLinealN(:,2:end)];
67
68 end
69
```

```
1 function [parametros,topologia]=generarPesosRolling(cuboDatosTemp,numEval)
2 %OBJETIVO: Extraer los parametros resultantes de la parte lineal como no lineal
3 %asi como de las neuronas.
4 %COMPORTAMIENTO: Dado el numero de evaluacion retorna los parametros y la
5 %topologia.
6 %RETORNA: Los vectores de parametros y la topologia.
7
8 parametros=cell2mat(cuboDatosTemp(numEval,6,1));
9
10 topologia=unirTopologia(cell2mat(cuboDatosTemp(numEval,11,1)));
11
12 end
13
14
```

```
1 function [parametros]=generarPesosIniciales(numeroCapasOcultas,numeroNeuronas,rezagosParteNoLineal,rezagosParteLineal,dentroMuestraNoLinealN,dentroMuestraLinealN)
2 %OBJETIVO: Generar los pesos sinapticos iniciales de la parte lineal como no lineal
3 %asi como la ponderacion de las neuronas.
4 %COMPORTAMIENTO: son capturadas las matrices que contienen informacion
5 %sobre los rezagos lineales y no lineales, el numero de neuronas.
6 %RETORNA: Las matrices phi, alpha y betha con los pesos iniciales.
7
8 %Obtener la cantidad de conecciones de la parte no lineal
9 numeroEntradas=size(rezagosParteNoLineal,2)+1;
10 %Pesos iniciales de la parte lineal se calculan MCO
11 phi=((dentroMuestraLinealN(:,2:end) '*dentroMuestraLinealN(:,2:end))^( -1))*(
(dentroMuestraLinealN(:,2:end) '*dentroMuestraLinealN(:,1));
12 %Generar los pesos para cada neurona con una distribucion uniforme entre -2 y 2
13 betha=-2+(2+2).*rand(1,numeroNeuronas);
14 %Generar los pesos para cada coneccion con una distribucion uniforme entre -2 y 2
15 alpha=-2+(2+2).*rand(1,numeroEntradas*numeroNeuronas);
16 %Transponer phi para que quede de la forma 1xX
17 phi=phi';
18 %Unir phi, alpha y betha para que quede un vector de 1xX
19 parametros=[phi alpha betha];
20
21 end
22
23
```

```
1 function llave=generarLlave(rezagosParteLineal,rezagosParteNoLineal,numeroNeuronas)
2 %OBJETIVO: Generar una llave que permita desagregar los pesos mas adelante.
3 %COMPORTAMIENTO: Adquiere el numero de rezagos y neuronas.
4 %RETORNA: Retorna un vector con la informacion suficiente para separar las matrices.
5
6 numeroEntradas=size(rezagosParteNoLineal,2)+1;
7 parametrosLineales=size(rezagosParteLineal,2)+1;
8 parametrosConecciones=parametrosLineales+numeroNeuronas;
9 parametrosNeuronas=numeroNeuronas;
10 llave=[parametrosLineales parametrosConecciones numeroEntradas];
11
12 end
```

```

1 function [dentroMuestraNoLineal, fueraMuestraNoLineal, dentroMuestraLineal, ↵
fueramuestraLineal]=generarInput(rezagosParteNoLineal,rezagosParteLineal,numHorizonte)
2 %OBJETIVO: Generar las matrices de diseño necesarias.
3 %COMPORTAMIENTO: Son capturadas las matrices donde se almacenan los rezagos de la ↵
parte
4 %lineal como de la no lineal, los datos y el numero de la columna que se usara.
5 %RETORNA: Las matrices de diseño necesarias.
6
7 global datos columnaSerie columnnaDesempeno tamanoHorizontes numeroHorizontes
8
9 %Obtener los valores que necesita la funcion
10 serie=datos(:,columnaSerie);
11 numRezagosParteNoLineal=size(rezagosParteNoLineal,2);
12 numRezagosParteLineal=size(rezagosParteLineal,2);
13 numPeriodos=size(datos,1);
14 numDentroMuestra=numPeriodos-tamanoHorizontes-(numeroHorizontes-numHorizonte);
15 numFueramuestra=numPeriodos-(numeroHorizontes-numHorizonte);
16 nuevosDatosParteNoLineal=zeros(numPeriodos,numRezagosParteNoLineal);
17 nuevosDatosParteLineal=zeros(numPeriodos,numRezagosParteLineal);
18
19 %Remplazar los valores dentro de la matriz vacia de la parte no lineal
20 for i=1:numRezagosParteNoLineal
21     for m=1:numPeriodos-rezagosParteNoLineal(1,i)
22         nuevosDatosParteNoLineal(m+rezagosParteNoLineal(1,i),i)=serie(m);
23     end
24 end
25
26 %Remplazar los valores dentro de la matriz vacia de la parte lineal
27 for i=1:numRezagosParteLineal
28     for m=1:numPeriodos-rezagosParteLineal(1,i)
29         nuevosDatosParteLineal(m+rezagosParteLineal(1,i),i)=serie(m);
30     end
31 end
32
33 nuevosDatosParteNoLineal=[serie(1:numFueramuestra,:) nuevosDatosParteNoLineal(1:↵
numFueramuestra,:)];
34 nuevosDatosParteLineal=[serie(1:numFueramuestra,:) nuevosDatosParteLineal(1:↵
numFueramuestra,:)];
35
36 % Crear 4 matrices de ceros para llenarlas con lo nuevos datos
37 dentroMuestraNoLineal=nuevosDatosParteNoLineal(1:numDentroMuestra,:);
38 fueraMuestraNoLineal=nuevosDatosParteNoLineal(numDentroMuestra+1:numFueramuestra,:);
39 dentroMuestraLineal=nuevosDatosParteLineal(1:numDentroMuestra,:);
40 fueraMuestraLineal=nuevosDatosParteLineal(numDentroMuestra+1:numFueramuestra,:);
41
42 %Organizar las matrices y les agrego el intercepto
43 dentroMuestraNoLineal= [dentroMuestraNoLineal(:,1) ones(size(dentroMuestraNoLineal,1),1) dentroMuestraNoLineal(:,2:end)];
44 dentroMuestraLineal= [dentroMuestraLineal(:,1) ones(size(dentroMuestraLineal,1),1) dentroMuestraLineal(:,2:end)];
45 fueraMuestraNoLineal= [fueramuestraNoLineal(:,1) ones(size(fueramuestraNoLineal,1),1) fueraMuestraNoLineal(:,2:end)];
46 fueraMuestraLineal= [fueramuestraLineal(:,1) ones(size(fueramuestraLineal,1),1) fueraMuestraLineal(:,2:end)];
47

```

48 end

49

```

1 function listo = generarGraficas(mejorANN, mejorENN, rezagosParteNoLineal, ↵
rezagosParteLineal, numHorizonte)
2 %OBJETIVO: Generar las graficas de las mejores redes.
3 %COMPORTAMIENTO: Las redes ingresan en una columna, se extrae los datos de cada una
4 %y se grafica.
5 %RETORNA: Devuelve las graficas y los valores para verificar las redes.
6
7 global datos columnaSerie columnaDesempeno tamanoHorizontes numeroHorizontes
8
9 numeroNeuronas=cell2mat(mejorANN(1,3))
10 numHorizonte=cell2mat(mejorANN(1,5))
11 phi=cell2mat(mejorANN(1,8))
12 betha=cell2mat(mejorANN(1,9))
13 alpha=cell2mat(mejorANN(1,10))
14 topologia=cell2mat(mejorANN(1,11))
15 rezagosParteLineal=cell2mat(mejorANN(1,12));
16 rezagosParteNoLineal=cell2mat(mejorANN(1,13));
17 tipoNormalizacion=1;
18
19 [dentroMuestraNoLineal, fueraMuestraNoLineal, dentroMuestraLineal, ...
20     fueraMuestraLineal]=generarInput(rezagosParteNoLineal, ...
21     rezagosParteLineal,numHorizonte);
22 [dentroMuestraNoLinealN, fueraMuestraNoLinealN, dentroMuestraLinealN, ...
23     fueraMuestraLinealN]=generarInputNormalizado(rezagosParteNoLineal, ...
24     rezagosParteLineal,numHorizonte,tipoNormalizacion);
25
26 %Ajuste ANN
27 A_1_AANN=dentroMuestraNoLinealN(:,2:end);
28 Z_2_AANN=A_1_AANN*(topologia(1:end-1,:).*alpha);
29 A_2_AANN=sigmoid(Z_2_AANN);
30 A_3_AANN=(dentroMuestraLinealN(:,2:end)*phi')+(A_2_AANN*((topologia(end:end,:)'.* ...
*beta')));
31
32 %Pronostico ANN
33 A_1_FANN=fueraMuestraNoLinealN(:,2:end);
34 Z_2_FANN=A_1_FANN*(topologia(1:end-1,:).*alpha);
35 A_2_FANN=sigmoid(Z_2_FANN);
36 A_3_FANN=(fueraMuestraLinealN(:,2:end)*phi')+(A_2_FANN*((topologia(end:end,:)'.* ...
*beta')));
37
38 %%%%%% ESPACIO PARA DESEMPEÑO %%%%%%
39
40 nDM=size(dentroMuestraLinealN,1);
41 nFM=size(fueraMuestraLinealN,1);
42
43 rmseDM=sqrt((1/nDM)*sum((dentroMuestraNoLinealN(:,1)-A_3_AANN).^2))
44 rmspeDM=sqrt((1/nDM)*sum(((dentroMuestraNoLinealN(:,1)-A_3_AANN). ...
/dentroMuestraNoLinealN(:,1)).^2));
45 maeDM=(1/nDM)*sum(abs(dentroMuestraNoLinealN(:,1)-A_3_AANN));
46 mapeDM=(1/nDM)*sum(abs((dentroMuestraNoLinealN(:,1)-A_3_AANN). ...
/dentroMuestraNoLinealN(:,1)));
47
48 rmseFM=sqrt((1/nFM)*sum((fueraMuestraNoLinealN(:,1)-A_3_FANN).^2))
49 rmspeFM=sqrt((1/nFM)*sum(((fueraMuestraNoLinealN(:,1)-A_3_FANN). ...
/fueraMuestraNoLinealN(:,1)).^2))

```

```

50 maeFM=(1/nFM)*sum(abs(fueraMuestraNoLinealN(:,1)-A_3_FANN))
51 mapeFM=(1/nFM)*sum(abs((fueraMuestraNoLinealN(:,1)-A_3_FANN)./fueraMuestraNoLinealN(:,1)))
52
53 %%%%%%%%%%%%%%
54
55 numeroNeuronas=cell2mat(mejorENN(1,3))
56 numHorizonte=cell2mat(mejorENN(1,5))
57 phi=cell2mat(mejorENN(1,8))
58 betha=cell2mat(mejorENN(1,9))
59 alpha=cell2mat(mejorENN(1,10))
60 topologia=cell2mat(mejorENN(1,11))
61 rezagosParteLineal=cell2mat(mejorENN(1,12));
62 rezagosParteNoLineal=cell2mat(mejorENN(1,13));
63
64 [dentroMuestraNoLineal, fueraMuestraNoLineal, dentroMuestraLineal, ...
65     fueraMuestraLineal]=generarInput(rezagosParteNoLineal, ...
66     rezagosParteLineal,numHorizonte);
67 [dentroMuestraNoLinealN, fueraMuestraNoLinealN, dentroMuestraLinealN, ...
68     fueraMuestraLinealN]=generarInputNormalizado(rezagosParteNoLineal, ...
69     rezagosParteLineal,numHorizonte, tipoNormalizacion);
70
71 %Ajuste ENN
72 A_1_AENN=dentroMuestraNoLinealN(:,2:end);
73 Z_2_AENN=A_1_AENN*(topologia(1:end-1,:).*alpha);
74 A_2_AENN=sigmoid(Z_2_AENN);
75 A_3_AENN=(dentroMuestraLinealN(:,2:end)*phi')+(A_2_AENN*((topologia(end:end,:)'.* ...
*beta')));
76
77 %Pronostico ENN
78 A_1_FENN=fueraMuestraNoLinealN(:,2:end);
79 Z_2_FENN=A_1_FENN*(topologia(1:end-1,:).*alpha);
80 A_2_FENN=sigmoid(Z_2_FENN);
81 A_3_FENN=(fueraMuestraLinealN(:,2:end)*phi')+(A_2_FENN*((topologia(end:end,:)'.* ...
*beta')));
82
83 %%%%%%%%%%%%%% ESPACIO PARA DESEMPEÑO %%%%%%%%%%%%%%
84
85 nDM=size(dentroMuestraLinealN,1);
86 nFM=size(fueraMuestraLinealN,1);
87
88 rmseDM=sqrt((1/nDM)*sum((dentroMuestraNoLinealN(:,1)-A_3_AENN).^2))
89 rmspeDM=sqrt((1/nDM)*sum(((dentroMuestraNoLinealN(:,1)-A_3_AENN). ...
/dentroMuestraNoLinealN(:,1)).^2));
90 maeDM=(1/nDM)*sum(abs(dentroMuestraNoLinealN(:,1)-A_3_AENN));
91 mapeDM=(1/nDM)*sum(abs((dentroMuestraNoLinealN(:,1)-A_3_AENN). ...
/dentroMuestraNoLinealN(:,1)));
92
93 rmseFM=sqrt((1/nFM)*sum((fueraMuestraNoLinealN(:,1)-A_3_FENN).^2))
94 rmspeFM=sqrt((1/nFM)*sum(((fueraMuestraNoLinealN(:,1)-A_3_FENN). ...
/fueraMuestraNoLinealN(:,1)).^2))
95 maeFM=(1/nFM)*sum(abs(fueraMuestraNoLinealN(:,1)-A_3_FENN))
96 mapeFM=(1/nFM)*sum(abs((fueraMuestraNoLinealN(:,1)-A_3_FENN). ...
/fueraMuestraNoLinealN(:,1)));
97

```

```
98 %%%%%%
99
100 figure(2)
101 subplot(1,2,1),
102 hold on
103 plot(dentroMuestraNoLinealN(:,1))
104 plot(A_3_AANN(:,1),'r')
105 plot(A_3_AENN(:,1),'g')
106 title('In Sample o Entrenamiento')
107 hleg1 = legend('DM','AANN','AENN');
108 hold off
109 subplot(1,2,2),
110 hold on
111 plot(fueraMuestraNoLinealN(:,1))
112 plot(A_3_FANN(:,1),'r')
113 plot(A_3_FENN(:,1),'g')
114 title('Out Sample o Pronostico')
115 hleg2 = legend('FM','FANN','FENN');
116 hold off
117
118 listo=1;
119
120 end
```

```

1 function [costo,topologia]=funcionCostoTopologia(parametros,poblacion,↖
dentroMuestraNoLinealN,dentroMuestraLinealN,tipofuncion,llave,numeroNeuronas,↖
numeroRezagos)
2 %OBJETIVO: Devolver el valor de la funcion de costo para unos parametros dados
3 %unas matrices de diseño Normalizadas, una topologia y se calculara en cierto
4 %tipo de funcion que se requiera.
5 %COMPORTAMIENTO: Segun el tipo de funcion, la topologia y los parametros separados de
6 %phi, alpha y betha se calcula el costo.
7 %RETORNA: El costo de la funcion.
8
9 %Decodifico los parametros
10 [parametros,poblacion]=decodificarParametros(poblacion,parametros,numeroRezagos);
11
12 %Recostruyo las matrices de parametros y la matriz de topologia
13 m=size(dentroMuestraNoLinealN,1);
14 [phi,betha,alpha]=separarMatrices(parametros,llave,numeroNeuronas);
15 topologia=separarTopologia(poblacion,llave,numeroNeuronas);
16
17 if tipofuncion==1
18     A_1=dentroMuestraNoLinealN(:,2:end);
19     Z_2=A_1*(topologia(1:end-1,:).*alpha);
20     A_2=sigmoid(Z_2);
21     A_3=(dentroMuestraLinealN(:,2:end)*phi')+ (A_2*((topologia(end:end,:)'.*↖
*betha')));
22     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001↖
*alpha(:,2:end).^2));
23     costo=((1/m)*sum((dentroMuestraNoLinealN(:,1)-A_3).^2))+reg+(0*mean(mean↖
(topologia)));
24 end
25
26 if tipofuncion==2
27     A_1=dentroMuestraNoLinealN(:,2:end);
28     Z_2=A_1*(topologia(1:end-1,:).*alpha);
29     A_2=sigmoid(Z_2);
30     A_3=(dentroMuestraLinealN(:,2:end)*phi')+ (A_2*((topologia(end:end,:)'.*↖
*betha')));
31     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001↖
*alpha(:,2:end).^2));
32     costo=((1/m)*sum(abs((dentroMuestraNoLinealN(:,1)-A_3)))+reg+(0*mean(mean↖
(topologia)));
33 end
34
35 if tipofuncion==3
36     A_1=dentroMuestraNoLinealN(:,2:end);
37     Z_2=A_1*(topologia(1:end-1,:).*alpha);
38     A_2=sigmoid(Z_2);
39     A_3=(dentroMuestraLinealN(:,2:end)*phi')+ (A_2*((topologia(end:end,:)'.*↖
*betha')));
40     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001↖
*alpha(:,2:end).^2));
41     costo=((1/m)*sum(1*(exp(-(dentroMuestraNoLinealN(:,1)-A_3))-(1*(-↖
(dentroMuestraNoLinealN(:,1)-A_3))-1))+reg+(0*mean(mean(topologia))));
42 end
43
44 if tipofuncion==4

```

```
45 A_1=dentroMuestraNoLinealN(:,2:end);
46 Z_2=A_1*(topologia(1:end-1,:).*alpha);
47 A_2=sigmoid(Z_2);
48 A_3=(dentroMuestraLinealN(:,2:end)*phi')+A_2*((topologia(end:end,:)'.*
*betha'));
49 reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001*
*alpha(:,2:end).^2));
50 costo=((1/m)*sum(0.3*(abs(dentroMuestraNoLinealN(:,1)-A_3)-(0.7*(
dentroMuestraNoLinealN(:,1)-A_3))))+reg+(0*mean(mean(topologia))));
```

51 end  
52  
53 end

```

1 function costo=funcionCosto(parametros,dentroMuestraNoLinealN,dentroMuestraLinealN,
2 tipoFuncion,llave,numeroNeuronas)
3 %OBJETIVO: Devolver el valor de la funcion de costo para unos parametros dados
4 %unas matrices de diseño Normalizadas y se calculara en cierto tipo de funcion
5 %que se requiera.
6 %COMPORTAMIENTO: Segun el tipo de funcion y los parametros separados de
7 %phi,alpha y betha se calcula el costo.
8 %RETORNA: El costo de la funcion.
9
10 m=size(dentroMuestraNoLinealN,1);
11 [phi,betha,alpha]=separarMatrices(parametros,llave,numeroNeuronas);
12
13 if tipoFuncion==1
14     A_1=dentroMuestraNoLinealN(:,2:end);
15     Z_2=A_1*alpha;
16     A_2=sigmoid(Z_2);
17     A_3=(dentroMuestraLinealN(:,2:end)*phi')+(A_2*betha');
18     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001
*alpha(:,2:end).^2));
19     costo=((1/m)*sum((dentroMuestraNoLinealN(:,1)-A_3).^2))+reg;
20 end
21
22 if tipoFuncion==2
23     A_1=dentroMuestraNoLinealN(:,2:end);
24     Z_2=A_1*alpha;
25     A_2=sigmoid(Z_2);
26     A_3=(dentroMuestraLinealN(:,2:end)*phi')+(A_2*betha');
27     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001
*alpha(:,2:end).^2));
28     costo=((1/m)*sum(abs((dentroMuestraNoLinealN(:,1)-A_3))))+reg;
29 end
30
31 if tipoFuncion==3
32     A_1=dentroMuestraNoLinealN(:,2:end);
33     Z_2=A_1*alpha;
34     A_2=sigmoid(Z_2);
35     A_3=(dentroMuestraLinealN(:,2:end)*phi')+(A_2*betha');
36     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001
*alpha(:,2:end).^2));
37     costo=((1/m)*sum(1*(exp(-(dentroMuestraNoLinealN(:,1)-A_3)))-(1*(-
(dentroMuestraNoLinealN(:,1)-A_3))-1)))+reg;
38 end
39
40 if tipoFuncion==4
41     A_1=dentroMuestraNoLinealN(:,2:end);
42     Z_2=A_1*alpha;
43     A_2=sigmoid(Z_2);
44     A_3=(dentroMuestraLinealN(:,2:end)*phi')+(A_2*betha');
45     reg = sum(0.01*sum(phi(:,2:end).^2)+sum(0.0001*betha(:,2:end).^2)+sum(0.0001
*alpha(:,2:end).^2));
46     costo=((1/m)*sum(0.3*(abs(dentroMuestraNoLinealN(:,1)-A_3)-(0.7*(
(dentroMuestraNoLinealN(:,1)-A_3)))))+reg;
47 end
48

```

49 end

```
1 function [parametrosPoblacionDec,poblacionDec]=decodificarParametros(poblacion, ↵
parametrosPoblacionCod,numeroRezagos)
2 %OBJETIVO: Decodificar la poblacion devolviendo un vector de parametros aumentado ↵
donde las unidades
3 %de la topologia que sean ceros sean recomuestas nuevamente como ceros.
4 %COMPORTAMIENTO: Recibe los parametros codificadas y recibe la topologia segun la
5 %topologia agrega ceros a los parametros no conectados.
6 %RETORNA: Parametros decodificados y la poblacion decodificada.
7
8 parametrosPoblacionDec=zeros(1,(numeroRezagos+1)+size(poblacion,2));
9 poblacionNueva=[ones(1,(numeroRezagos+1)) poblacion];
10
11 for i=1:size(poblacionNueva,2)
12     if poblacionNueva(1,i)==1
13         indicador(1,i)=i;
14     end
15 end
16
17 indicador=indicador(indicador ~= 0);
18
19 for ii=1:size(parametrosPoblacionCod,2)
20
21     if indicador(1,ii) ~= 0
22         parametrosPoblacionDec(1,indicador(1,ii))=parametrosPoblacionCod(1,ii);
23     end
24
25 end
26
27 poblacionDec=poblacion;
28
29 end
```

```
1 function [parametrosPoblacionCod,poblacionCod]=codificarParametros(poblacion,↖
parametrosPoblacion,numeroRezagos)
2 %OBJETIVO: CODificar un vector de parametros devolviendo el vector reducido
3 %donde todos las unidades de la topologia que sean ceros sean eliminadas.
4 %COMPORTAMIENTO: Recibe los parametros y recibe la topologia segun la
5 %topologia elimina los parametros no necesarios.
6 %RETORNA: Parametros codificados y la poblacion que se uso para la codificacion.
7
8 for i=1:numeroRezagos+1
9
10    indicador(i)=i;
11
12 end
13
14
15 for i=1:size(poblacion,2)
16
17    if poblacion(1,i)==1
18        indicador(i+numeroRezagos+1)=i+numeroRezagos+1;
19        indicador=indicador(indicador ~= 0);
20    end
21
22 end
23
24 parametrosPoblacionCod=parametrosPoblacion(1,indicador);
25 poblacionCod=poblacion;
26
27 end
```

```

1 function [parametrosMejorTopologia,mejorTopologia,costoMejorTopologia,↖
ultimaGeneracion,cuboDesempeno] = algoritmoEvolutivo(parametrosIniciales,↖
topologiaInicial,topologia,numeroCapasOcultas,rezagosParteNoLineal,rezagosParteLineal,↖
dentroMuestraNoLinealN,dentroMuestraLinealN,fueraMuestraNoLinealN,fueraMuestraLinealN,↖
dentroMuestraNoLineal,dentroMuestraLineal,fueraMuestraNoLineal,fueraMuestraLineal,↖
tipoFuncion,numeroNeuronas)
2 %OBJETIVO: El algoritmo genetico realiza evaluacion del desempeño, cruce
3 %mutacion hasta el numero de generaciones dado.
4 %COMPORTAMIENTO: Remitirse a un texto especializado
5 %RETORNA: Retorna los parametros de la mejor topologia, la mejor Topologia,
6 %el costo de esta una matriz mostrando la ultima generacion y un cubo mostrando
7 %la evolucion entre generaciones.
8
9 %Define los parametros del Algoritmo Genetico
10 numeroBits=size(topologia,1)*size(topologia,2);
11 numeroCromosomas=50;
12 numeroGeneraciones=4;
13 numeroPadres=2;
14 tasaMutacion=0.01;
15 numeroMutaciones=tasaMutacion*numeroBits*(numeroCromosomas-1);
16
17 %Crea el numero de cromosomas aleatorios para la topologia
18 poblacion=round(rand(numeroCromosomas,numeroBits));
19
20 %Crea un numero de pesos iniciales para la primera generacion
21 for numCromosomas=1:numeroCromosomas
22
23     parametrosPoblacion(numCromosomas,:)=generarPesosIniciales(numeroCapasOcultas,↖
numeroNeuronas,rezagosParteNoLineal,rezagosParteLineal,dentroMuestraNoLinealN,↖
dentroMuestraLinealN);
24     llave=generarLlave(rezagosParteLineal,rezagosParteNoLineal,numeroNeuronas);
25
26 end
27
28 %Inserta topologias y parametros
29 poblacion(1,:)=ones(1,numeroBits);
30 poblacion(2,:)=topologiaInicial;
31 parametrosPoblacion(2,:)=parametrosIniciales;
32
33 %Crea matrices vacias y obtiene informacion para el Algoritmo Genetico
34 numeroRezagos=size(rezagosParteLineal,2);
35 costo=zeros(numeroCromosomas,1);
36
37 %Comienza el Algoritmo Genetico
38
39 for ib=1:numeroGeneraciones
40
41     for i=1:numeroCromosomas
42
43         %Para optimizar los pesos
44         %Codifica
45         [parametrosPoblacionCod,poblacionCod]=codificarParametros(poblacion(i,:),↖
parametrosPoblacion(i,:),numeroRezagos);
46         %Optimiza
47         options=optimset('HessUpdate','bfgs','LargeScale','off','MaxFunEvals',↖

```

```

1000000,'MaxIter',10000);
48      [x,fval]=fminunc(@(parametrosPoblacionCod) funcionCostoTopologia↖
(parametrosPoblacionCod,poblacionCod,dentroMuestraNoLinealN,dentroMuestraLinealN,↖
tipoFuncion,llave,numeroNeuronas,numeroRezagos),parametrosPoblacionCod,options);
49      %Decodifica
50      [parametrosPoblacionDec,poblacionDec]=decodificarParametros(poblacion(i,:),↖
x,numeroRezagos);
51      parametrosPoblacion(i,:)=parametrosPoblacionDec;
52      %Obtiene las medidas de desempeño
53      [rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM] =↖
obtenerDesempeno(parametrosPoblacion(i,:),dentroMuestraLinealN,dentroMuestraNoLinealN,↖
fueraMuestraLinealN,fueraMuestraNoLinealN,poblacion(i,:),llave,numeroNeuronas);
54      desempenoFM=((rmseFM+rmspeFM+maeFM+mapeFM)/4);
55      %Guarda el desempeño por individuo
56      desempeno(i,:)={[desempenoFM},{parametrosPoblacion(i,:)},{{poblacion(i,:)}},];
57      %Guarda el desempeño por individuo entre las generaciones
58      cuboDesempeno(i,:,ib)=[i,ib,{desempenoFM},{parametrosPoblacion(i,:)},↖
{poblacion(i,:)}];
59
60    end
61
62    %Organiza los mejores cromosomas por su desempeño
63    [desempeno,ind]=sortrows(desempeno,1);
64    poblacion=poblacion(ind(1:numeroPadres),:);
65    parametrosPoblacion=parametrosPoblacion(ind(1:numeroPadres),:);
66
67    %Pone las ristras que seran cruzadas
68    cruce=ceil((numeroBits-1)*rand(numeroCromosomas-numeroPadres,1));
69    cruceLineal=ceil((numeroRezagos-1)*rand(numeroCromosomas-numeroPadres,1));
70
71    %Obtiene la descendencia de los padres
72    for ic=numeroPadres+1:2:numeroCromosomas
73
74        %Obtiene la topologia de los hijos
75        poblacion(ic,1:cruce)=poblacion(1,1:cruce);
76        poblacion(ic,cruce+1:numeroBits)=poblacion(2,cruce+1:numeroBits);
77        poblacion(ic+1,1:cruce)=poblacion(2,1:cruce);
78        poblacion(ic+1,cruce+1:numeroBits)=poblacion(1,cruce+1:numeroBits);
79        %Obtiene los parametros de los hijos en la parte no lineal
80        parametrosPoblacion(ic,numeroRezagos+1:numeroRezagos+cruce)↖
=parametrosPoblacion(1,numeroRezagos+1:numeroRezagos+cruce);
81        parametrosPoblacion(ic,numeroRezagos+cruce+1:end)=parametrosPoblacion(2,↖
numeroRezagos+cruce+1:end);
82        parametrosPoblacion(ic+1,numeroRezagos+1:numeroRezagos+cruce)↖
=parametrosPoblacion(2,numeroRezagos+1:numeroRezagos+cruce);
83        parametrosPoblacion(ic+1,numeroRezagos+cruce+1:end)=parametrosPoblacion(1,↖
numeroRezagos+cruce+1:end);
84        %Obtiene los parametros de los hijos en la parte lineal
85        parametrosPoblacion(ic,1:cruceLineal)=parametrosPoblacion(1,1:cruceLineal);
86        parametrosPoblacion(ic,cruceLineal+1:numeroRezagos)=parametrosPoblacion(2,↖
cruceLineal+1:numeroRezagos);
87        parametrosPoblacion(ic+1,1:cruceLineal)=parametrosPoblacion(2,1:↖
cruceLineal);
88        parametrosPoblacion(ic+1,cruceLineal+1:numeroRezagos)=parametrosPoblacion(1,↖
1,cruceLineal+1:numeroRezagos);

```

```

89
90     end
91
92     %Muta solo a los hijos y si no es la ultima generacion
93     if ib~=numeroGeneraciones
94         for ic=1:numeroMutaciones
95             %Muta la topologia
96             ix=ceil((numeroPadres)+(numeroCromosomas-(numeroPadres))*rand);
97             iy=ceil(numeroBits*rand);
98             poblacion(ix,iy)=1-poblacion(ix,iy);
99             %Muta los pesos en la parte no lineal
100            ix=ceil((numeroPadres)+(numeroCromosomas-(numeroPadres))*rand);
101            iy=ceil((numeroRezagos)+(numeroBits)*rand);
102            parametrosPoblacion(ix,iy)=(-2+(2+2).*rand);
103        end
104    end
105 end
106
107
108 %Obtengo la ultima generacion obtengo su desempeño
109 for ii=1:numeroCromosomas
110
111     [rmseDM,rmspeDM,maeDM,mapeDM,rmseFM,rmspeFM,maeFM,mapeFM] = obtenerDesempeno(
112     (cell2mat(desempeno(ii,2))),dentroMuestraLinealN,dentroMuestraNoLinealN,↖
113     fueraMuestraLinealN,fueraMuestraNoLinealN,poblacion(ii,:),llave,numeroNeuronas);
114     desempenoFM=((rmseFM+rmspeFM+maeFM+mapeFM)/4);
115     [parametrosPoblacionCod,poblacionCod]=codificarParametros(poblacion(ii,:),
116     parametrosPoblacion(ii,:),numeroRezagos);
117     costo=funcionCostoTopologia(parametrosPoblacionCod,poblacion(ii,:),↖
118     dentroMuestraNoLinealN,dentroMuestraLinealN,tipofuncion,llave,numeroNeuronas,↖
119     numeroRezagos);
120     almacenDatos(ii,:)=[ii,desempenoFM,costo,{poblacion(ii,:)},desempeno(ii,2)];
121     ultimaGeneracion=almacenDatos;
122
123 ultimaGeneracion=sortrows(ultimaGeneracion, 2);
124 resultadoOpt=ultimaGeneracion(1,:);
125 parametrosMejorTopologia=cell2mat(resultadoOpt(1,5));
126 mejorTopologia=cell2mat(resultadoOpt(1,4));
127 mejorTopologia=separarTopologia(mejorTopologia,llave,numeroNeuronas);
128 costoMejorTopologia=cell2mat(resultadoOpt(1,3));
129
130 end

```